UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

PROYECTO FIN DE CARRERA

SOFTWARE FOR THE VISUALIZATION OF BLOOD FLOW VELOCITY VECTORS FROM AN ULTRASOUND SCANNER

INGENIERÍA DE TELECOMUNICACIÓN

Carlos Moreno López

Mayo 2015

SOFTWARE FOR THE VISUALIZATION OF BLOOD FLOW VELOCITY VECTORS FROM AN ULTRASOUND SCANNER

AUTOR: Carlos Moreno López
TUTOR: Jørgen Arendt Jensen
PONENTE: Jesús Bescós Cano

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo 2015

Resumen

La creciente importancia de la ultrasonografía en el estudio del flujo de sangre ha propiciado que múltiples investigadores trabajen en pos de avances en este campo. Se ha constatado que los investigadores en este área dedican una cantidad de tiempo significativa a visualizar sus resultados. Por lo tanto, la existencia de una herramienta que pudiera realizar esta labor por ellos sería extremadamente valiosa para facilitar el progreso de sus investigaciones. Habiéndose identificado MATLAB como una herramienta de software matemático ampliamente utilizada por los investigadores tanto en las universidades como en la industria, este proyecto abarca el desarrollo de un programa escrito en MATLAB que permita a los investigadores, que trabajan en la estimación de la velocidad del flujo sanguíneo, visualizar sus resultados. El programa implementado tiene dos scripts de visualización principales: uno para las visualizaciones en 2D y otro para visualizaciones en 3D, que es el futuro en este campo. Un grupo de funciones auxiliares, gobernado a través de las opciones del programa, permite a los investigadores realizar visualizaciones con cada script bajo diferentes parámetros. Tras haber sido desarrollado, el código ha sido analizado, concluyéndose con una evaluación positiva de su funcionamiento. Asimismo, el programa ya ha sido utilizado satisfactoriamente por investigadores en el campo.

Palabras clave: *ultrasonidos, velocidad, sangre, visualización, 2D, 3D, programa, matlab*

Abstract

The growing importance of ultrasound imaging systems in the study of blood flow has led multiple researchers to investigate techniques which expand this field. It has been found that the researchers in this field spend a significant amount of time in visualizing their results. Therefore, the existence of a tool that could perform this task for them would be extremely valuable for facilitating the progress of their investigations. Having identified MATLAB as a numeric computation program widely used by researchers both in universities and the industry, this project encompasses the development of a program written in MATLAB which allows the researchers in blood flow velocity estimation to visualize their results. The implemented program has two main visualization scripts: one for 2D visualizations and another for 3D visualizations, which is the future of this field. A group of auxiliary functions, governed through the program options, allow the researchers to perform visualizations under different settings with each script. After its development, the tool has been analysed, concluding in a positive assessment of its functioning. Furthermore, the program has already been successfully tested by researchers.

Keywords: *ultrasounds, velocity, blood, visualization, 2D, 3D, program, matlab*

Acknowledgements

In this lines I want to express my gratitude to the people that have made possible this thesis. Firstly, I want to thank Professor Jørgen Arendt Jensen for giving me the opportunity of working in the Center for Fast Ultrasound Imaging, a group of excellent researchers and great people. I appreciate a lot the guidance he has provided me with during the past months and the experience I have won by working with him. I also want to thank Michael Johannes Pihl for giving me his confidence since the beginning, guiding me in my first steps in the researching group. The realization of the project would not have been possible without the great advice and supervision provided by Simon Holbek in the final part of the work. Apart from my supervisors, the whole group has provided me with invaluable experience through enriching discussions and cooperation. I want to thank specially the collaboration of Carlos Armando Villagómez Hoyos, but also Jacob Bjerring Olesen, David Bradway and Morten Fischer Rasmussen.

I would like to thank the work of the teachers from Universidad Autónoma de Madrid that have participated in my development as a person and engineer. I am specially grateful to Jesus Bescós for accepting to guide me in this project from the distance. I also want to thank Irene González-Aller Zavala for her marvellous work in the International Relations Office.

This project would not have been possible without the support of a family to be proud of. Thanks to my parents Carlos and Maribel, and to my sister, Elena. Sometimes words are not enough to express a feeling.

I also consider Carlos Sanchez Bueno as part of my family, I really appreciate the university years of hard work and good times that we have had, but also the all the previous ones where we have been friends. This gratitude should be extended to the rest of my fellows in the university and the students association AET. Hugo Peire, Sergio de la Cruz and Adrian Cobos. Thank you. To all my friends in the university, but also outside it. To all my friends in Spain, but also all over the world, and specially here in Denmark. Thanks Murray Meissner. Thanks Muriel González. Thanks Sandra Torres, it has been an unbelievable journey. And specially, thanks to Helia Relaño Iborra. In these last lines I want to express the inexpressible gratitude for all your support and advice during the last months, being there every time I needed you.

To all of you that have made this project possible, sincerely: Gracias.

Carlos Moreno López. Lyngby, Denmark. May 5th, 2015.

Table of Contents

Resumen **I**

Abstract **III**

Acknowledgements **V**

List of Figures **XI**

List of Tables **XII**

Glossary **XIII**

1 Introduction **1**
1.1 Motivation 2
1.2 Structure 3

2 Context **5**
2.1 Circulatory system 5
2.2 Cardiovascular imaging procedures 7
2.3 Blood velocity estimation using Doppler Effect 9
2.4 Blood flow visualization with commercial ultrasound scanners 11

3 Design **15**
3.1 Analysis 15
3.1.1 Main specifications 15
3.1.2 Design discussion 16
3.2 Design of the implementation 17
3.3 Data discussion 18
3.3.1 Researching scanner 19
3.3.1.1 2D data 20
3.3.1.2 3D data 21
3.3.2 Commercial scanners 23

4 Implementation **25**
4.1 Implementation process 25
4.2 Description of the program's modules 27

4.2.1 Two dimensions visualization . . . 27
4.2.1.1 Input . . . 27
4.2.1.2 Default . . . 28
4.2.1.3 Load Data . . . 28
4.2.1.4 Prepare Data . . . 28
4.2.1.5 Mask . . . 33
4.2.1.6 Visualization 2D . . . 36
4.2.1.7 Calculate velocity range and/or vectors maximum . . . 51
4.2.2 Three dimensions visualization . . . 52
4.2.2.1 Input . . . 53
4.2.2.2 Default . . . 53
4.2.2.3 Load Data . . . 54
4.2.2.4 Prepare Data . . . 54
4.2.2.5 Mask . . . 55
4.2.2.6 Calculate velocity range and/or vectors maximum . . . 55
4.2.2.7 Visualization 2D . . . 55
4.2.2.8 Visualization 3D . . . 55
4.3 Description of the program's structures . . . 61
4.3.1 Data . . . 61
4.3.1.1 2D data . . . 62
4.3.1.2 3D data . . . 63
4.3.2 Options . . . 63
4.3.2.1 General . . . 63
4.3.2.2 Geometry . . . 65
4.3.2.3 Display . . . 65
4.3.2.4 Frames . . . 65
4.3.2.5 Mask . . . 65
4.3.2.6 Code . . . 65

5 Evaluation **67**
5.1 Testing . . . 67
5.2 Output analysis . . . 67
5.2.1 2D output . . . 68
5.2.2 3D output . . . 70
5.3 Processing time analysis . . . 71
5.3.1 2D visualization . . . 71
5.3.2 3D visualization . . . 72

6 Conclusions and Work Ahead **75**
6.1 Future work . . . 76

References **79**

Appendices **83**

A Introducción **85**
A.1 Motivación . . . 86

A.2 Estructura 87

B Conclusiones y trabajo futuro **89**

B.1 Trabajo futuro 90

C Presupuesto **93**

D Pliego de condiciones **95**

List of Figures

1.1 Diagram of the role of this project in the researchers working pipeline. 2

2.1 Diagram showing the blood flow in the human body. 6
2.2 Recreation of an electrocardiogram. 7
2.3 Model of the ultrasound's interaction with blood. 10
2.4 SIEMENS ACUSON P300 Ultrasound Scanner visualization. 12
2.5 BK Medical Pro Focus Ultraview Scanner visualization. 13

3.1 Diagram of the program's pipeline. 16
3.2 Schematic diagram of the program's organization. 18
3.3 Photographs of the back and front of SARUS with a 2D transducer connected. 20
3.4 Diagram that illustrates the acquisition of a synthetic aperture image. 21

4.1 Diagram of the program's first code pipeline. 25
4.2 Diagram of the iterative design methodology followed during the program's implementation. . 26
4.3 Diagram of the pipeline followed by the 2D visualization script. 27
4.4 Diagram of the B-mode matrix formation (before interpolation). 28
4.5 Diagram of the B-mode scan conversion and visualization. 29
4.6 B-mode matrix before the scan conversion. 29
4.7 B-mode matrix after the scan conversion. 30
4.8 B-mode image in gray scale. 30
4.9 Simplified diagram that illustrates the scan conversion process. 31
4.10 Diagram of the pipeline followed by the 3D data processing. 32
4.11 B-mode image interpolated in a certain region. 32
4.12 Vessel visualization that illustrates the difference between the sizes of velocity and static data. 33
4.13 Message boxes requesting one or more than one polynomials to create the mask. 34
4.14 Image models for the mask creation. 35
4.15 Polynomial function delimiting the mask. 35
4.16 Basic visualization of the velocity data (colormap). 37
4.17 Velocity data visualized through vectors and colormap. 38
4.18 Velocity data visualized through streamlines and colormap. 38
4.19 Visualization of the velocity in one point against time, using an auxiliary plot. 39
4.20 Visualization of the velocity values in one segment of the image, using an auxiliary plot. . . . 40
4.21 Representation of the values of a matrix's row from the B-mode plus velocity image matrix. . 43

4.22 Velocity visualization of static data. 46
4.23 Velocity visualization of flow data with only positive values. 46
4.24 Velocity visualization of flow data with only negative values. 47
4.25 Velocity visualization of flow data with both positive and negative values. 47
4.26 Velocity against time auxiliary plot in three different frames of the same dataset. 49
4.27 Velocity profile auxiliary plot in three different frames of the same dataset. 50
4.28 Two cross planes visualized in a 3D space. 52
4.29 Diagram of the pipeline followed in the 3D visualization script. 53
4.30 3D visualization of two cross planes. 57
4.31 3D visualization of two cross planes with vectors representation. 57
4.32 Diagram that illustrates the coordinate system of the 3D environment's *view* in MATLAB. . . . 58
4.33 Three different frames of a *gif* that turns the view around a 3D visualization. 60
4.34 Diagram of the fields of Data structure. 62
4.35 Diagram of the fields of Options structure. 64

5.1 Elements of the basic visualization of the velocity data (colormap). 68
5.2 Elements of the velocity data visualized through vectors and colormap. 68
5.3 Elements of the velocity data visualized through streamlines and colormap. 69
5.4 Elements of the visualization of the velocity in one point against time. 69
5.5 Elements of the visualization of the velocity values in one segment of the image. 70
5.6 Elements of the 3D visualization of two cross planes. 70
5.7 Elements of the 3D visualization of two cross planes with vectors representation. 71

A.1 Diagrama del rol de el presente proyecto en la proceso de trabajo de los investigadores. 86

List of Tables

3.1 Example of the file composition of a 3D velocity measurement outputted by SARUS. 22

5.1 2D visualization script processing time relation between its sections. 72
5.2 3D visualization script processing time relation between its sections. 72

Glossary

Abbreviations

1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
CT	Computed Tomography
EBCT	Electron-Beam Computed Tomography
PET	Positron emission tomography
SPET	Single Photon Emission Tomography
SPECT	Single Photon Emission Computed Tomography
NMR	Nuclear Magnetic Resonance
MRI	Magnetic Resonance Imaging
US	Ultrasound
M-mode	Motion mode
B-mode	Brightness mode
GB	Gigabyte
RAM	Random-Access Memory
RF	Radio Frequency
TO	Transverse Oscillation
SA	Synthetic Aperture
SARUS	Synthetic Aperture Real-time Ultrasound System
CFU	Center for Fast Ultrasound Imaging
DTU	Danmarks Tekniske Universitet

CHAPTER 1

Introduction

The average human body contains around 5.5 litres of blood [1]. This amount of blood is carried through the body by the circulatory system, which has the responsibility of providing oxygen and nutrients to the organs and of disposing carbon dioxide and metabolic end products that the cells expel.

The importance of the circulatory system for the human health leads to a specific medical speciality to cover the issues related to the vascular medicine, this speciality is called angiology. Consequently, the vascular medicine needs appropriate tools to diagnose possible diseases in the circulatory system.

In this context, the relevance of ultrasound imaging systems in the study of issues in the circulation of the blood through the vessels has to be highlighted. Ultrasound imaging systems are considered a riskless medical imaging system, which combined with its well appreciated non-invasive characteristic, makes them desirable for the study of blood flow. This is done by using the Doppler Effect, as can be seen in [18]. The information about the blood flow allows doctors to detect certain diseases and, thus, act accordingly. Furthermore, it has to be taken into account that blood flow velocities change with both time and space dependence [4], so the estimation of the velocity space components simultaneously becomes a desirable goal for the current researches in this field.

The development of new velocity estimation methods and the improvement of the existing ones have the final objective of yielding more precise images of the blood flow and its complex patterns. During the development of these methods, an immediate visualization of the estimated data would facilitate the evaluation of the improvements achieved by this researchers. The mentioned immediate visualization should be seen as part of the development of the estimation methods, and previous to the clinical image yielding that is done in the hospitals.

This said, the aim of this thesis corresponds to the design and implementation of a tool that addresses an illustrative fast visualization of the blood flow velocity. A diagram that illustrates this procedure can be seen in figure 1.1. The blood flow velocity data is obtained with different velocity estimators, working in various medical ultrasound scanners with variable equipment and parameters. Because of that, the raw velocity data can have a wide variety of formats, and there is no homogeneous way of visualizing it. Therefore, the existence of a tool that overcomes this situation becomes a need. On this basis, the work developed in this thesis must be seen as a pure engineering project that faces a existing need in order to solve it.

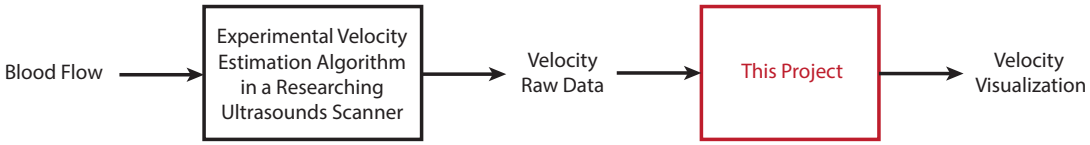


Figure 1.1 – Diagram of the role of this project in the researchers working pipeline.

This report covers, first, the background information required for the comprehension of the developed project, which is explained in depth in the remaining chapters. The objectives pursued in this research will be addressed in the present chapter with the aim of giving the reader a overall vision of the investigation.

1.1 Motivation

The main goal of this thesis project comes from the visualization needs of the researchers working in the improvement of the blood flow velocity calculation techniques. As it is explained in chapter 2, the cardiovascular system is a complex network of vessels in charge of the transport of the blood through the body. Due to the complexity of the blood flow dynamics through the human body and its interaction with the ultrasound fields, several techniques has been proposed in order to overcome the different challenges of this field of study [26]. Researchers working in the development of these techniques need to visualize the results achieved in their investigations, being able to evaluate the improvements accomplished and share their advances.

The objective of this project is to develop a tool that provides a helpful method of visualization of the velocity vectors obtained by researchers.

With the aim of being more specific, the nature of the tool that solve such a need should be found. According to The Columbia Encyclopedia [33], an ordered sequence of computational instructions necessary to achieve a solution is called computational program. Which means that the tool to solve the proposed situation, computationally, can be referred as program from now on in the present thesis report.

In order to facilitate the researchers work, the development of such program should be carried out in their data process environment. Having identified MATLAB as a numeric computation program widely used by the researchers both in universities and the industry [16], the design of a visualization tool based in the programming language of this software comes as a logical consequence.

As is stated in [17], word for word: *"Matlab is a data analysis, prototyping and visualization tool with built in support for matrices and matrix operations, excellent graphics capabilities, and a high-level programming language and development environment"*. However, the great potential of this tool could be diminished by the complex use of some of its utilities. For of this reason, the desired visualization tool is meant to be an easy way to generate fast visual representations of the researchers results, avoiding the loose of time and effort in this part of the researching process.

It can be concluded that the objective of this project is to design and develop a MATLAB program that visualizes the data obtained by researchers in blood flow velocity estimation. The development of this program implies the development of several functionalities related to the data interpolation, vectors processing, image treatment and video generation.

1.2 Structure

This thesis report is divided in six chapters. The following points give a short description of each chapter after this introductory one:

- **Chapter 2** contextualizes the thesis, going through the human cardiovascular system and the medical imaging systems that have been used for its visualization. The state of the art of the blood flow visualization with ultrasounds is also analysed in this chapter.
- **Chapter 3** describes the process that leaded to the program's current design, going from an analysis of its specifications to the discussion about the design and implementation. The nature of the data faced by the program is also discussed in this chapter.
- **Chapter 4** gives the reader an overview of the program's implementation, exposing also its functionalities while describing its modules.
- **Chapter 5** explains how the program's evaluation has been carried out, addressing also the analysis of the results of the program's execution testing.
- **Chapter 6** summarizes the conclusions reached in this investigation and presents the possibilities for further development of the work carried out.

CHAPTER 2

Context

With the objective of contextualizing this investigation, this chapter presents a brief description of the human circulatory system and its functioning. Moreover, a summary of the evolution of the use of medical imaging systems in the study of the cardiovascular system is introduced and, finally, visualization methods applied in the field of blood flow velocity estimation will be reviewed.

2.1 Circulatory system

The presence of the circulatory system in the human body comes as a consequence of the evolution process of the human being. This means that, as a complex multicellular organism, a person's body cannot transport substances between cells using just diffusion as is done between adjoining cells. In order to perform the movement of substances between the different regions of the body, humans have the heart, the blood vessels and the blood itself. Those elements form the cardiovascular system, that combined with the lymphatic system constitute the circulatory system [6].

As it is explained deeply in [1] and [5], the main role of the circulatory system is to provide every part of the body with the nutrients that the cells need for their correct functioning, as well as collecting the waste products delivered by the same cells. Those nutrients and waste products are transported through the body dissolved in a liquid called plasma. Formed elements (cells and cell fragments) are suspended in this plasma, composing what is known as the blood. There are two kinds of blood cells: the red blood cells (erythrocytes) and the white blood cells (leukocytes), but only one kind of cell fragments, that are the platelets.

Blood circulates through the body thanks to a system of interconnected tubes known as the vascular system. This system is formed by the blood vessels (arteries, veins and capillaries), which are organized in two circuits both originating and terminating in the heart. Figure 2.1 shows a schematic representation of this structure. The pumping action of the heart produces a difference of pressures in the vessels that enable the rapid flow of blood throughout the body.

The blood is carried away from the heart in vessels called arteries, those vessels have very flexible walls that allow them to contract and expand as a result of every pulsation of the blood. On the other hand, vessels called veins carry the blood from different parts of the body towards the heart. Those vessels diameter is larger than the corresponding arteries and have thinner and less elastic walls to let the blood flow back to the heart, as is detailed in [2].

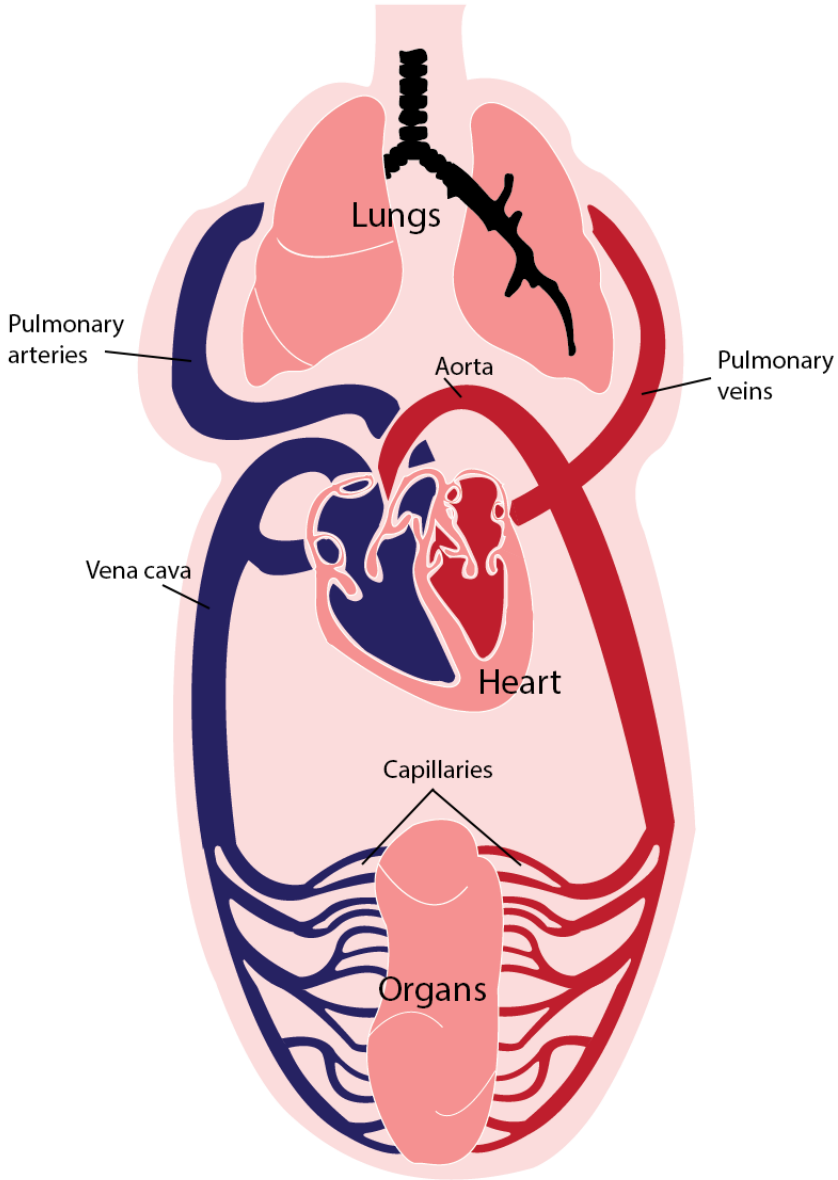


Figure 2.1 – Diagram showing the blood flow in the human body. As it is explained in [1]: *"the cardiovascular system forms a closed loop, so that blood pumped out of the heart through one set of vessels returns to the heart by a different set"*. There are two circuits: the pulmonary circulation (from the heart through the lungs and then back to the heart) and the systemic circulation (from heart through all the organs and tissues of the body except the lungs and then back to the heart).

Blood vessels are widely branched through the body, forming a network that connects the heart with the smallest branches, called capillaries. Thus making possible that almost every cell is less than a few cell diameters away from one termination of this vessels set.

The pumping behaviour of the blood flow implies that the velocity of the blood flow at one certain point of the vessels network is not constant in time, exhibiting a pulsatile flow pattern. This fact, combined with the

blood flow's variable parabolic shape profile of the flow (due to the viscosity of the fluid), turns the study of blood fluid dynamics into a quite complex field. Figure 2.2 illustrates a recreation of an electrocardiogram, that is a visualization of the electrical signals that rule the heartbeat activity and, thus, the pulsatile flow pattern. For further information about the heart pumping mechanism, and how it affects the blood flow, [1] and [7] can be consulted.

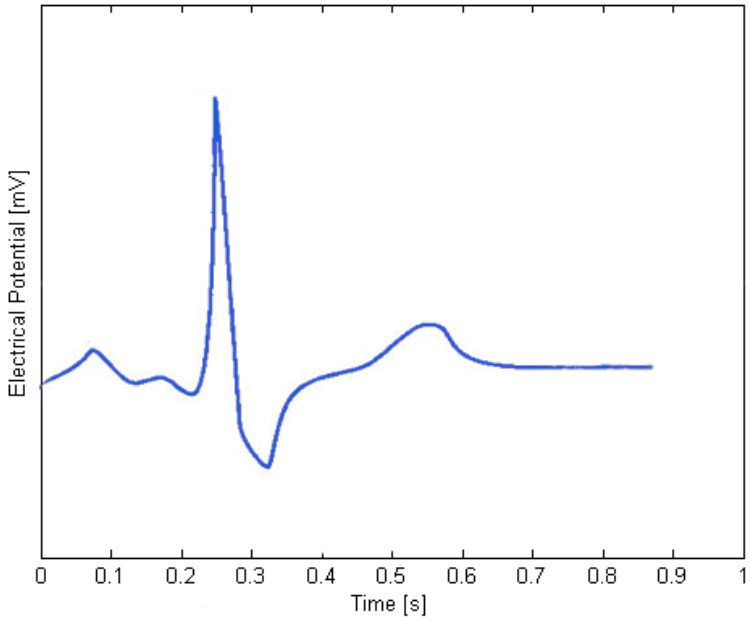


Figure 2.2 – Recreation of an electrocardiogram, which presents the potential of the electrical events of the heart against time. The pulsatile flow pattern of the blood flow comes from this pumping behaviour.

As is widely treated in the literature, there are many diseases of the cardiovascular system that can affect the already complex blood circulation. This makes necessary a study of the blood flow as well as the estimation of its velocity, in order to make possible the detection of malfunctions in this significant and complex group of organs from the human body.

2.2 Cardiovascular imaging procedures

The main challenge faced by the cardiovascular imaging systems is the existence of time dimension as one of the main parameters in the study of the circulatory system [8]. This leads to the evaluation of the main imaging tools according to, not only their spatial and density resolution, but their temporal resolution too [9].

As it is recognized through the literature, the main medical imaging methods are: radiology, magnetic resonance imaging, nuclear medicine imaging and ultrasound imaging. All these tools are defined by the physical properties they use for the image generation and also attending to which relevant information can be obtained from each of them. For a deeper knowledge about the medical imaging systems reference [3] can be consulted.

The first technique applied successfully to medical imaging was the use of X-rays, which can provide planar images of the inside of the human body, according to the characteristic attenuation coefficient of the different tissues. The first radiography attempt dates from 1895, but it was not until the decade of 1910 when X-rays were improved to yield better resolutions that permitted the study of the cardiovascular system [19]. In 1929 the use of a liquid radiocontrast agent in the radiographic cardiac examination introduced the angiocardiography as an efficient cardiovascular imaging system [11].

Concurrently with the beginnings of the radiography, the Lumière brothers developed the cinematograph [20]. With the development of both this technology and the X-rays imaging, the first cineangiocardiography was performed in 1939, introducing the time parameter to the angiocardiography [19].

Based in the X-rays principle but rotating the X-ray tube around the patient, the Computed Tomography (CT) has also been applied to study of the cardiovascular system. The first clinical CT was installed in a hospital in 1971, but it was not until 1979, with the introduction of the Electron-Beam Computed Tomography (EBCT), that cardiovascular CT imaging could get rid of motion artifacts caused by the pumping of blood [12] [19].

In parallel with radiology, nuclear cardiology has been developed since the first attempts with radiation which took place in the 1920s. In relation to the topic of this thesis it must be highlighted the development of a radiotracer technique to measure blood flow velocity in 1926, by H. Blumgart [14]. In the following years, a slow succession of improvements in the radioisotopes and detection techniques took place, as well as the advance in the time-activity caption. It was not until the decade of 1950 when the Positron Emission Tomography (PET) and the Single Photon Emission Tomography (SPET) were developed, [15] can be consulted for a deeper knowledge of those advanced nuclear imaging techniques, and how cardiovascular clinical images are obtained from tracers in the blood flow.

On the other hand, attending to the magnetic property of the water molecules existing in the human body, Magnetic Resonance Imaging (MRI) was developed. Dating from the early 1950's [13], the Nuclear Magnetic Resonance (NMR) phenomenon created the foundations of the MRI, but it was two decades later, in 1973, when the first two-dimensional Magnetic Resonance Images were published. However, during the 1970 decade the acquisition time was too long to avoid the time-changing pumping characteristic of the heart [19], this was solved in the 1980s. The application of MRI techniques to the study of cardiovascular system is, then, a logical consequence from its capability to difference between different soft tissues, as well as the possibility of flow visualization and quantification [12].

Ultrasound (US) imaging is recognized as the medical imaging method that meant the biggest advantage for cardiovascular diagnostic since the X-rays. Even though ultrasounds were discovered in the 18th century, it was not until many years later when their use was related to the medicine diagnosing field. The advances in the piezo-electric materials research and its application to ultrasounds generation made possible the appearance of US in medical diagnosis, in 1941 [19]. However, the application of this technique to cardiovascular diagnose was achieved in the 1950s with the first echocardiograms, and later on, in the early 1960s with the beginning of the M-mode echocardiographys originated by the M-mode (motion mode) recording of US echoes [21]. Once the time parameter was included in the Ultrasound imaging, a significant growth in the progress of echocardiography was achieved with the appearance and development of real-time two-dimensional echocardiography in the decades of 1960 and 1970.

In the early second half of the 20th century, another great advance in the field of US imaging was accomplished making use of a concept from the previous century, the Doppler Effect. This mathematical relationship explain the change in frequency of a wave for an observer moving relative to its source [22]. The application of the Doppler Effect to US enabled the measurement of red blood cells' velocity and thus, the velocity of blood flow. The combination of this technique with US imaging originated the duplex ultrasonography in 1974, becoming a meaningful clinical advance [19]. Three years later, in 1977, Bernouilli fluid dynamics equation was brought to the blood flow study, establishing a relationship between the velocity of the blood flow and the pressure it exerts to the vessel walls. After a succession of improvements in the flow visualization, US imaging systems have become the most complete and comprehensive cardiac diagnostic modality, providing real-time visual information with a non-invasive system [23].

Even though some of the previously mentioned imaging systems can provide convenient visualization characteristics, the number of advantages that US imaging can offer has positioned it as a valuable diagnostic tool in several medical disciplines, cardiology among others. The high use of US imaging systems is mainly due to its rapid image formation (real time imaging capability) and its non-invasive nature, but also because of its cost-effectiveness in comparison to other medical imaging systems [23]. Due to all those reasons, researching in US imaging field can be considered a reliable way to give beneficial advances to the medical community, and thus, society.

2.3 Blood velocity estimation using Doppler Effect

After having introduced the Doppler Effect as the physical concept behind the measurement of blood flow velocity with ultrasounds, a simplified explanation of the theory that involves this phenomenon is given in this section.

The Doppler Effect is described as the the change in frequency of a periodic event, perceived by a target, due to a relative movement of the target and the event's source. For an easy comprehension of the phenomenon, this concept can also be explained related to the time variable (in this case the period of the periodic event), knowing the relation between those two variables:

$$T = \frac{1}{f} \tag{2.1}$$

Where T corresponds to the period and f to the frequency of the mentioned event.

Taking as reference the simple model of ultrasound's interaction with blood that can be observed in figure 2.3, the study of this interaction can be performed using the Doppler Effect related to time, identifying the transducer (ultrasonic probe) as a static target and the blood as the source in motion.

It may seem unclear why the blood is considered the source in this model, but a simplified explanation of the ultrasound imaging process can clarify it. The element that commonly generates and receives the ultrasound signal in this kind of clinical imaging is called transducer. Working as emitter, the transducer generates an ultrasound wave, which produces scatterers in its interaction with the blood. Through the reception of these echoes (scatterers) the US imaging system generates the image. [2] Thus, in this situation, the blood acts as an emitter of the echoes that the transducer receives (working as receptor). Taking into account that the blood is flowing through the vessels, the same particle of the blood will emit scatterers from two different

positions for two subsequent wave emissions, which reveals the motion characteristic of the source.

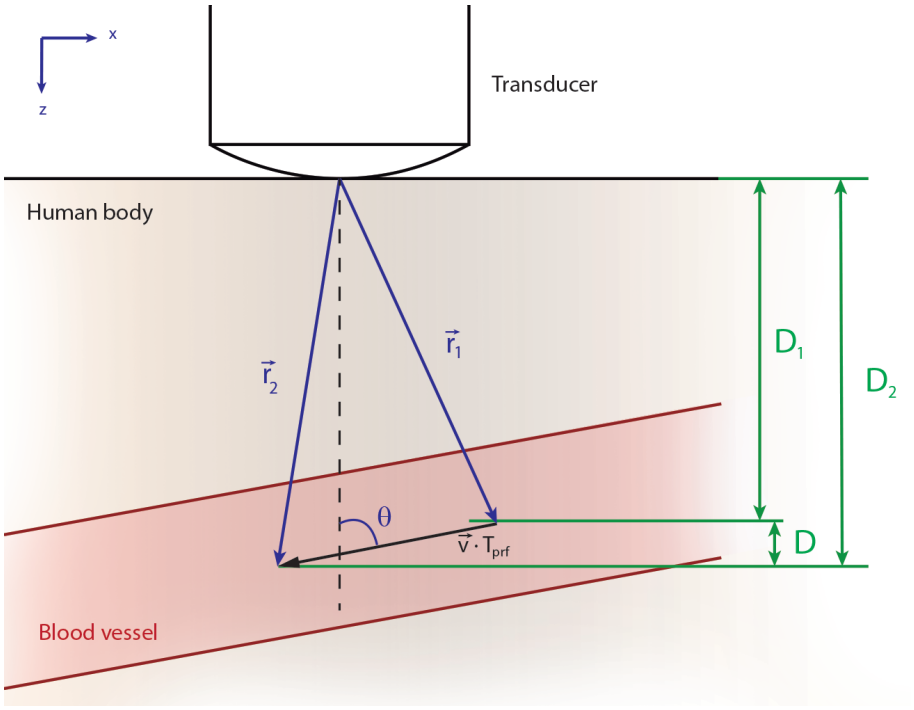


Figure 2.3 – Model of the ultrasound's interaction with blood. The transducer is placed against the skin, under which there is a blood vessel (red). The vector of direction of the blood flow ($\vec{v}T_{prf}$) has been represented in black, inside the blood vessel.

Velocity is commonly defined as *"the change in displacement with respect to time"* [29], which can be written as:

$$v = \frac{\Delta x}{\Delta t} \tag{2.2}$$

Where the velocity v corresponds to the relation of Δx and Δt, space displacement and a time shift, respectively.

Thus, the relationship between the vectors of figure 2.3 can be synthesised in the following equation:

$$|\vec{r}_2| = \vec{v}T_{prf} + |\vec{r}_1| \tag{2.3}$$

Where T_{prf} represents the time period between two subsequent wave emissions, being PRF a reference to the pulse repetition frequency. Also, $|\vec{r}_1|$ and $|\vec{r}_2|$ are the vectors between the transducer and the points where the wave hit the blood producing scattering, and $\vec{v}$ is the vector that represents the velocity of the blood flow in this certain direction.

Applying basic trigonometry [30] to the velocity vector, its z component v_z can be defined as:

$$v_z = |\vec{v}|\, cos\theta \tag{2.4}$$

Where $|\vec{v}|$ is the module of the velocity vector $\vec{v}$, and θ is the angle of this vector with the ultrasound beam.

On the other hand, taking the distances D_1 and D_2 as the z components of $|\vec{r}_1|$ and $|\vec{r}_2|$, distance D corresponds to the difference between them:

$$D = D_1 - D_2 \tag{2.5}$$

Where D can be defined using equation 2.2 to state the relationship between distance (D), velocity (v_z) and time (T_{prf}). Applying also the equality presented in equation 2.4:

$$D = v_z T_{prf} = |\vec{v}|\, T_{prf} cos\theta \tag{2.6}$$

Furthermore, the difference in time (relative to the pulse emission) between the arrival of scatterers from the two subsequent emissions is defined by the following equation [2]:

$$t_s = \frac{2D}{c} \tag{2.7}$$

Where this difference in time is called t_s, being c the speed of sound and maintaining the notation for D.

Finally, substituting D from equation 2.6, the time shift between the two received signals (t_s) can be written as:

$$t_s = \frac{2\,|\vec{v}|\, T_{prf} cos\theta}{c} \tag{2.8}$$

Which can also be presented in terms of frequency:

$$f_d = \frac{2\,|\vec{v}|\, f_0 cos\theta}{c} \tag{2.9}$$

Where f_d is the Doppler frequency and f_0 is the center frequency of the signal emissions.

The module of velocity can be then presented as a function of known values:

$$|\vec{v}| = \frac{t_s c}{2T_{prf} cos\theta} = \frac{f_d c}{2 f_0 cos\theta} \tag{2.10}$$

It is important to bear in mind, however, that beyond the simplified scheme presented in this section, the use of ultrasound imaging systems to estimate blood flow velocity is a complex field that has grown through different velocity estimation techniques. More information about velocity estimation using the Doppler Effect can be found in the literature: detailed definitions of the estimation techniques appear in [31], among others. An approach to the estimation of more than one velocity components of blood flow can be found in [26].

2.4 Blood flow visualization with commercial ultrasound scanners

Currently, almost all the commercial (medical) ultrasound scanners are able to provide Color Doppler imaging, it should be highlighted that this flow visualization system has proven a great value for the clinical assessing of blood flow. This imaging technique consists in the superposition of colour-coded maps of velocity on top of grey-scale visualizations of tissue anatomy [25]. The estimation of velocity that allows this visualization is performed by the application of the Doppler Effect, giving its name to the imaging technique. Typically, the flow towards the transducer is presented in a red scale of colors, meanwhile a blue scale of colors depicts the flow away from the transducer. Figure 2.4 shows the screenshot of a commercial scanner

providing a Color Doppler visualization of blood flow.

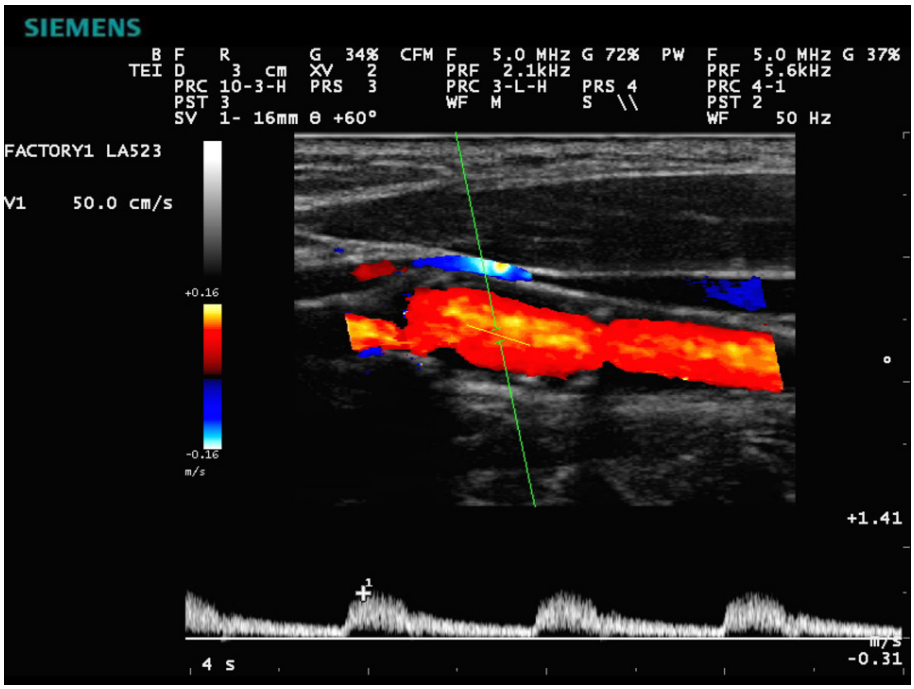


Figure 2.4 – Carotid artery and jugular vein visualization using the SIEMENS ACUSON P300 Ultrasound Scanner, applying Color Doppler Imaging. Image from [45].

As pointed out before, the complexity of the process of blood flow velocity estimation exceeds the model presented in the previous section. Color Doppler imaging is based in the velocity estimation for a number of subsequent scan lines of the transducer, being necessary to estimate the phase shift between consecutively received lines. Autocorrelation and cross-correlation algorithms present considerable advantages to estimate the derivative of this phase, being the autocorrelation the most widely used for color flow imaging in commercial scanners [25].

It must be emphasized that the conventional velocity estimators in ultrasound are capable of estimating only the velocity component that is parallel to the ultrasound beam (axial velocity component v_z) [26]. This estimation is appropriate assuming that the flow is parallel to the vessel, but as it has been shown in section 2.1 of this chapter, vessels are not long and straight, or always healthy, either.

Although several techniques have been proposed in order to find a velocity estimation beyond the velocity component parallel to the ultrasound beam, the Transverse Oscillation (TO) method has received more of the author's attention in the development of this thesis. This has been motivated by the prolific research of the Center for Fast Ultrasound Imaging (CFU) in the topic [27], being CFU the researching group in which this project has been carried out.

The TO method estimates both the axial and the transverse components of flow velocity. To understand the functioning of this velocity estimation method, it must be reminded that the oscillations in the transmitted ultrasonic pulse are what allows the estimation of the axial component of velocity in section 2.3. By introducing a known transverse oscillation in the US field, the received signals will depend on this transverse

oscillation, being possible to also estimate the transverse component of the velocity.

Figure 2.5 shows the application of Transverse Oscillation method on a commercial scanner. What is clearly observable in the combined B-mode and velocity image is that velocity vectors have been depicted both with arrows and a colormap, illustrating a velocity estimation beyond the v_z component.

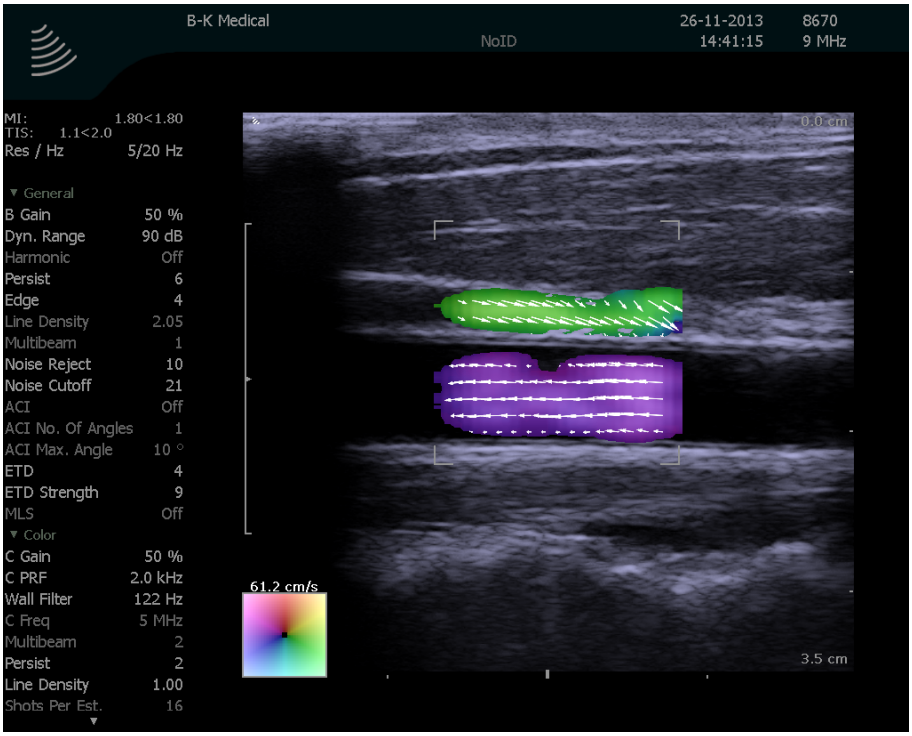


Figure 2.5 – Carotid artery and jugular vein visualization using the BK Medical Pro Focus Ultraview Scanner with a linear array transducer, applying the Transverse Oscillation (TO) method for Vector Flow Imaging. Image courtesy of David Bradway.

CHAPTER 3

Design

This chapter provides a general overview of the design process that leaded to the development of the program that fulfils the desired requisites. Through these pages the objectives followed are analysed together with the specifications and limitations of the program.

3.1 Analysis

The aim of this project is the design and development of a MATLAB program for the researchers that study the blood flow velocity using ultrasounds. The main objective of the project is, thus, that the program end up being used by researchers to facilitate their investigations.

The development of this project has been carried out in the Center for Fast Ultrasound Imaging (CFU), which means that the data used for the program development has been provided by researchers from this group. The final result is the outcome of a process of code developing based on fixed objectives, but improved by the group's researchers feedback during the program implementation.

3.1.1 Main specifications

The designed tool must fulfil the following specifications:

- The tool should be able to provide 2D and 3D visualization of velocity vectors.
- The tool should be a program developed in MATLAB as it is the researchers working environment.
- The tool should be able to manage different data input.
- The tool should be adaptable to the evolving needs of the researchers.

As it can be observed in figure 3.1, the tool pursued is a program coded in MATLAB that process velocity data estimated by the researchers in both commercial and researching ultrasound scanners, with variant complexity depending on the source. After receiving and processing the velocity data the program must be able to visualize it according to a number of options, in order to fit each researcher's visualization needs.

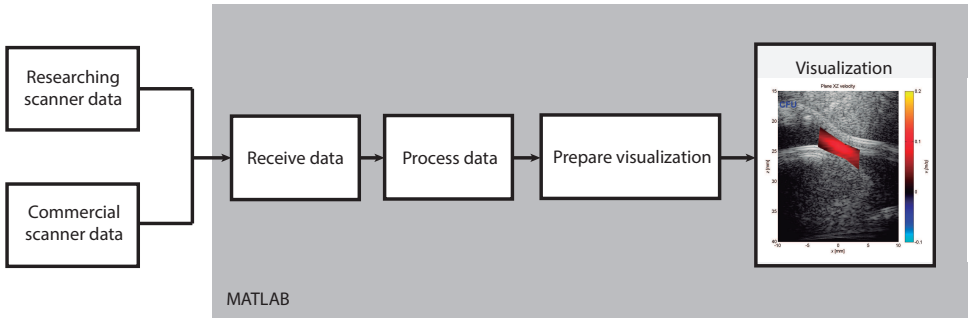


Figure 3.1 – Diagram of the program's pipeline. The data is obtained from an ultrasound scanner, either a commercial or a researching one. Then it is loaded in the program developed in MATLAB, that process and visualize it.

3.1.2 Design discussion

Having these specification statements as foundation, the design of the tool has taken into consideration the suggestions of the researchers, that demand simplicity, quality and velocity of the visualization. One of the main challenges addressed in the design of the tool has been to find a balance between an illustrative visualization and a short processing time to achieve it, without forgetting that the developed software had to be easy to use.

However, easy of usage is a very relative characteristic that depends on the knowledge of the user. In this case, the tentative users of the program are researchers on blood flow study with assumed high MATLAB skills. Based on that, the visualization software developed has been done through a group of organized and commented scripts and functions, not having a visual interface, but giving the users the possibility of adapting the code to future research.

Having this first design decision taken, the next subject matter addressed has been the need of a fast visualization. This issue has been faced in two levels: the number of steps executed in the visualization process and the use of the RAM memory during the execution of the program. Taking into account that parts of the visualization process related to the data loading and processing can be avoided in consecutive visualizations of the same data, the visualization scripts have been designed as a pipeline process divided in code sections, letting the user execute only the code sections needed in subsequent visualizations after the first one. On the other hand, the use of RAM memory has been taken into account in the implementation of the code assuming that it would presumably face data sizes that can exceed the capacity of a regular RAM memory installed in the researchers computers from CFU (between 2 and 8 Gigabytes).

Meanwhile, the meaning of an *illustrative visualization* must be clarified in order to explain the design decisions taken on its behalf. According to the definition of *illustrative* given in the Oxford Dictionary of English [32], an illustrative visualization of the velocity of blood flow would be a visualization that serves as an explanation of how the blood flows in the studied area according to the estimation of the researchers. The term "explanation" implies a clear visualization easy to recognise, documented with the needed information in order to provide the information without any ambiguity. For this reason, the units of al the values depicted should be represented as well as graphic information that would facilitate the understanding of what is being visualized.

In the pursuit of an illustrative visualization, another significant decision should be highlighted: Taking into account that the program is conceived to be able to represent any velocity component through colors of the image (its colormap), it was decided to show the positive values in a warm range of values, from dark to light, and the negative values with a cold one, using dark colors for small negative velocities and light ones to high negative values. This color code for the velocity representation has been chosen following the industry's unwritten agreement regarding Color Doppler visualization (as shown in section 2.4), adapting it to the researchers feedback. Moreover, having as reference the discussed scanners that implement arrows to show the vectors of velocity in 2D, this visualization concept has been included in the implementation of the vectors depiction.

In spite of all the discussed characteristics, the author (agreeing with CFU) considered that one more attribute was needed to make the program an useful instrument: the capability of performing a visualization without any adjustment needed in the code. This means that the program must count with a strong set of default settings that support the possible data sources providing an direct visualization. In addition to that, the program must be able to offer the user a wide group of visualization options easily modifiable. This two last characteristics, combined with all the previous ones, would make this group of scripts a desirable visualization tool, and thus, used by the researchers, which is the final goal of this project.

3.2 Design of the implementation

Having the analysed statements clear in mind, the design of the program structure can be addressed. The following is a breakdown of requisites together with the main premises fixed as desirable objectives:

- MATLAB program
- Fast visualization
- Good default settings
- Easy use for researchers
- Illustrative visualization
- Acceptance of multiple input data
- Multiple visualization options
- 2D and 3D visualization possibility
- Program adaptability

In order to develop a program that fulfils those objectives and requisites, the code has been divided in two main scripts that call functions for specific tasks of the visualization process. The reason behind the separation of the program in two scripts is the differentiation between the visualization procedures of 2D and 3D data. Through this pair of scripts the program provides two main visualization environments: A two dimension plane visualization and a three dimension visualization of two crossed planes. A diagram that illustrates this organization can be observed in figure 3.2.

Furthermore, the functioning of the program is based in two MATLAB data structures that work as a vehicle to share the information between the different functions called in each visualization module. These two structures are called `data` and `options`, referencing to what is saved in each of them: the data to visualize and the options that specify how to visualize it.

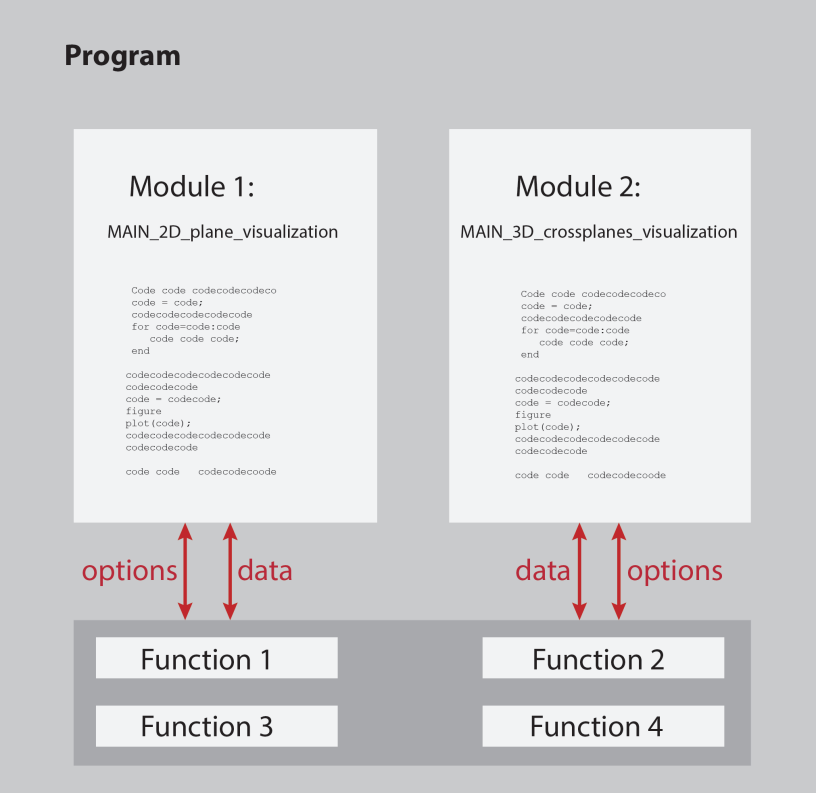


Figure 3.2 – Schematic diagram of the program's organization. The two visualization scripts (Module 1 and Module 2) have been depicted as rectangles, and the structures that carry the information inside the program (options and data) have been presented as double-headed arrows. The functions have been represented by four rectangles, however, the real number of written functions called by the scripts rises to more than 40.

After having presented the chosen composition of the visualization program, a review of the data sources for the program should be presented. Once this is done, next chapter will address the implementation of the program's code.

3.3 Data discussion

As has been already pointed out, the project has been developed inside the researching group CFU, working alongside with its researchers in order to provide them with an illustrative tool to visualize their achievements in the field of blood velocity estimation. Therefore, the target program has been designed in order to operate with the data estimated by the current researchers in this researching group, without forgetting the fact that it

should be easily adaptable to new data sources from other researchers.

This section provides a full description of the data sources supported by the developed program. The visualization of such data provides the researchers an useful way to evaluate: the velocity data estimation methods, the techniques of data processing after its acquirement and the data measured itself. Focusing in one or another depending on the focus of their study at each point of their investigation.

The data handled by the researchers in the Center for Fast Ultrasound Imaging for the study of the blood flow comes from different sources depending on the ultrasounds scanner used, the transducer, or the data-obtaining method, among others.

The sources of data considered in this project can be grouped in two primary categories: one referred to an experimental researching scanner designed and developed by CFU and another that consist of commercial scanners from the industry. The following presents a classification of the different data sources attending to its acquisition.

3.3.1 Researching scanner

The researching ultrasounds scanner that has been considered in this project is the one developed by the Center for Fast Ultrasound Imaging from the Electrical Engineering department of the Technical University of Denmark. This scanner, called SARUS (Synthetic Aperture Real-time Ultrasound System), was described in 2013 as *"the first of its kind in the world and the only instrument yet created that can display the movement of blood in 3D"* [34].

The construction of such scanner was initiated in 2005 [35], wiling to *"handle advanced imaging concepts"*. As it is exposed in [28], this unique experimental scanner *"is capable of transmitting and receiving ultrasound signals for 1024 independent transducer elements simultaneously"*, while commercial transducers typically operate 192. Being able to receive the Radio Frequency (RF) signal on all the transducers in parallel at a sampling frequency of 70 MHz, with 12 bits precision, the signals *"can be stored in real time for several seconds"* [28], saving the data as a set of time frames. Employing *"sixty-four 1-Gbit/s links and four 10-Gbit/s links"* [28], SARUS is connected to a Linux computer cluster for *"fast data storage and off-line processing"* [28]. Photographs of the back and front of SARUS are shown in figure 3.3.

Moreover, this scanner is able to implement real-time Synthetic Aperture (SA) processing [36], which is an imaging technique used for increasing the *"resolution and contrast of ultrasound images"* [37] [38]. In fact, *"in traditional ultrasound systems a trade-off is made between the range of detectable velocities and the number of estimation points"* [39], but using SA for vector velocity estimation overcomes this issue.

Thanks to the aforementioned characteristics, this unique instrument allows the members of CFU to take their studies to unreachable limits for other researchers in the fields of ultrasonic flow estimation and ultrasound imaging. On the other hand, due to its complexity, SARUS has to be programmed specifically for each measurement, providing the information acquired by each transducer element together with all the specifications about the scanner calibrations for the measurement. The scanner software *"is written in C++ and runs under Matlab for high level access to the system"* [38].

Figure 3.3 – Photographs of the back (left image) and front (front image) of SARUS with a 2D transducer connected. Images courtesy of Morten Fischer Rasmussen and Michael Johannes Pihl.

In addition to the 2D measurements, SARUS is capable of measuring the blood flow in one more dimension through the 3D Transverse Oscillation. This method allows the estimation of the three components of velocity, calculating the two transverse velocity components through two doubleoscillating fields [26].

Due to the described capability of SARUS, it can be expected to receive two different data types from this source: 2D data and 3D data.

3.3.1.1 2D data

The 2D data from SARUS utilized along this this thesis project corresponds to measurements performed with both linear and phased array transducers (multi-element transducers). However, the data used mainly comes from measurements performed using the linear array transducer BK8670, from BK Medical. Information about the characteristics of the basic multi-element transducers can be consulted in [2].

Both B-mode and velocity data matrices are generated by SARUS, applying different procedures. As mentioned before, SARUS employs SA imaging, which yield a high resolution image from the summation of each full low-resolution image received after every pulse emission. The low-resolution images derive from the RF data from the transducer elements. For a deeper understanding of the procedure, figure 3.4 illustrates a 2D imaging acquisition. More information about this process can be found in [38]. Data is saved in sets of one to several frames depending on the time lapse of the measurement and its sampling period. The number of time frames faced in the development of this project in the 2D datasets goes from 1 to 235 frames depending on the dataset.

Although the velocity estimation is performed in SARUS with the corresponding algorithm implemented by each researcher in the system, the data received in each measurement needs to be processed to be operational. Velocity data was obtained both in polar and cartesian coordinates, depending on the estimation algorithm.

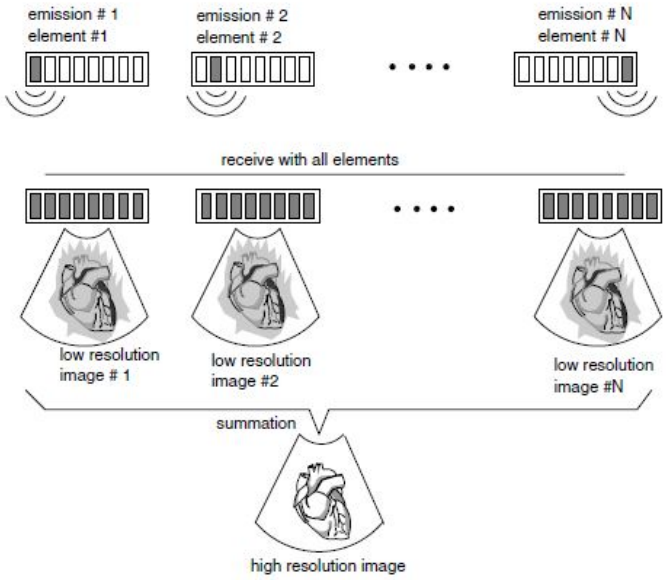


Figure 3.4 – Diagram that illustrates the acquisition of a synthetic aperture image (from [38], originally from [40]). Each time, only a single element is used in transmission, creating a cylindrical wave. All the elements are used in reception. The received signals originated by the receive echo, that comes from all directions, are used to create a low quality image. The high resolution image is created by summing them.

3.3.1.2 3D data

As part of his PhD research, Michael Johannes Pihl developed a method for estimating 3D velocities using a Transverse Oscillation approach [26]. His implementation of this method in SARUS allows the measurement of the three components of blood flow velocity with the scanner.

Although the use of SARUS is not limited to one specific transducer [26], a 2D array transducer is needed to perform 3D measurements. The data used in this project comes from measurements performed using a phased array 2-D transducer fabricated by Vermont S.A., using a matrix of 32x32 active elements out of the 35x32 available in the transducer. The 1024 elements of the 32x32 matrix are connected to the corresponding 1024 channels available on SARUS in order to obtain the data. It should also be mentioned that the center frequency of this transducer is 3.5 MHz.

Independent B-mode and velocity data files are generated by SARUS for each frame. The velocity estimation is performed inside the scanner, through the algorithm implemented for the measurement, in this case the method used would be the 3D Transverse Oscillation. The data saved for each time frame comes in a *".mat"* file that needs to be processed. The number of time frames handled in this thesis in the 3D datasets goes from 1 to 87 frames. The size of these one-frame velocity files was, in average, in the order of 0.2 GB, what means that data sets with more than 10 frames should be handled carefully in order to avoid problems with the RAM of the researchers' computers. To give the reader an overview of the nature of the file, table 3.1 presents the fields right under the data structure of one time frame of a 3D velocity measurement (the B-mode data is stored in another data file). Almost all the fields presented in the table contain their own sub-fields.

Table 3.1 – Example of the file composition of a 3D velocity measurement outputted by SARUS.

Name	Value
PD	<1818x18 double>
PD_temp	<1818x1 double>
P_rcv	<18x32 double>
P_rcv_mf	<18x32 double>
Q	<1x18 cell>
RF	<1x1 struct>
bfm_rf_3d	<4-D double>
bmode	<1818x18 double>
choose_data_set	18
choose_seq_no	5
data_path	<1x18 cell>
disc	<1x18 cell>
do	<1x1 struct>
ems	<1x18 cell>
flow_angle	<1x18 cell>
i	18
iii	5
par	<1x1 struct>
par1	<1x1 struct>
sarus	<1x1 struct>
sys	<1x1 struct>
tid	<18x5 double>
v_abs	<1818x18 double>
v_abs_center	<18x5 double>
v_abs_max	<18x5 double>
v_abs_mean	<18x5 double>
v_max	1.4025
vx_TO	<1818x18 double>
vx_actual	<1818x1 double>
vx_center	<18x5 double>
vy_TO	<1818x18 double>
vy_actual	<1818x1 double>
vy_center	<18x5 double>
vz_TO_zx	<1818x18 double>
vz_TO_zy	<1818x18 double>
vz_actual	<1818x1 double>
vz_ax	<1818x18 double>
vz_center	<18x5 double>
xmt	<1x1 struct>

In November of 2013, Simon Holbek started his PhD research continuing part of the research performed by Michael Johannes Pihl before him. His progress in the field resulted in new 3D data measurements. For a better understanding on how the blood flow 3D velocity vectors are measured with SARUS, Simon's publication [41] can be consulted.

3.3.2 Commercial scanners

The Center for Fast Ultrasound Imaging has among its facilities several commercial US scanners, mainly from BK Medical, which has an ongoing collaboration with the researching center. Although the visualization program developed in this thesis project handled data from all the different scanners available in CFU, it should be mentioned that the most used commercial scanner source during the program's development was a BK Medical 2202 Profocus Ultraview scanner with a linnear array BK8670 transducer. The research interface UA2227 allows the data extraction from the scanner. More information about this process can be found in [42].

The data from the commercial scanners was manipulated by the researchers before being received by this thesis' author. The datasets have from one to several 2D frames of independent B-mode and velocity matrices that needed to be adapted to the visualization program, depending on the alteration of the data performed by the researchers. Velocity data was given in both cartesian and polar coordinates.

CHAPTER 4

Implementation

The objective of this chapter is to present and analyse the technical decisions taken through the implementation of the designed program, summarizing some of the problems faced during this process. It will also be explained how the main modules have been structured and how the communication within the different functions is performed.

4.1 Implementation process

The implementation of the pursued tool started with the programming of the code for performing a 2D visualization of a plane of B-mode and blood flow data. The program started, then, with a main function for visualizing 2D data, calling a number of functions for the different functionalities needed.

The program evolved through the enlargement of the main 2D function, together with the development of needed functions. Later, another main function was developed, in order to perform the 3D visualizations. The program structure can be observed in the already discussed figure 3.2.

When developing the 2D visualization main function, the first data considered was a frame data measured with SARUS by the researcher Michael Johannes Pihl. The data provided was already processed and corresponded to a 2D B-mode plane and the two corresponding inplane vectors of velocity, combined in the same matrix. In this first case, the program pursued pipeline presented in figure 3.1, was simplified to the functions presented in figure 4.1.

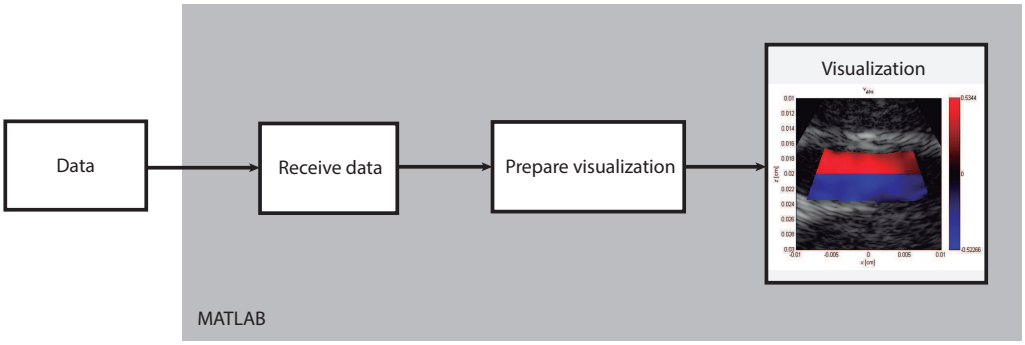


Figure 4.1 – Diagram of the program's first code pipeline. The data is loaded in the program developed in MATLAB and visualized with it.

The visualization of the provided data was adapted to the basic researchers needs, setting a starting point for the subsequent improvements. Being in this initial stage, two different tasks appeared as potential improvements to the basic code: adapting the program to any CFU data input and upgrading the program to several visualization options.

Having chosen to adapt the program to any data source managed by the researchers in the group as second step, different functions were created to adapt the data from each specific data source to the program data structure. This would allow to visualize the different data with the same functions independently of the source. This step, however, was challenging. The high number of differences between the group different data sources increased significantly the complexity of the program implementation. Also, the correct interpolation of all the different data required numerous coding attempts.

The implementation of the additional visualization features was an elaborated process where the communication with the researchers was key. The implementation of each feature followed a retro-alimented loop where the the feedback from the researchers leaded to a number of implementation iterations. Figure 4.2 presents a diagram that illustrates the procedure followed. The iterative design improved both the user interface and the functionalities of the program. The words of Jakob Nielsen can be quoted to emphasize the importance of the researchers participation in the software development process: *"Because even the best usability experts cannot design perfect user interfaces in a single attempt, interface designers should build a usability engineering life cycle around the concept of iteration. Iterative development of user interfaces involves steady design refinement based on user testing and other evaluation methods."* [10].

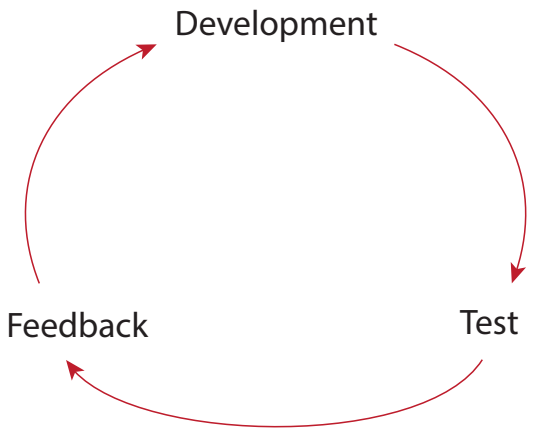


Figure 4.2 – Diagram of the iterative design methodology followed during the program's implementation.

In order to facilitate the use of the program, a strong set of default settings is needed. The implementation of these default settings involved, again, the researchers feedback and the testing of several methods in order to find the optimum one. The fact that the same code have to deal with a wide number of different data sources and researcher's necessities suppose an steamed increment of its complexity. Making the program robust and easy to use also implied a great number of challenges and implementation iterations.

Providing the user with information about the steps followed in the program execution and the state of the data visualization was also considered a relevant feature. Such code was, thus, also implemented.

As one of the last steps of the program implementation, the development of the error control through the code was performed. Also, an exhaustive commenting of the code was needed with the intention of allowing the researchers to adapt the code to new demands when if it became needed.

The implemented code was, however, reviewed a considerable number of times with the aim of increasing the efficiency of its performance.

4.2 Description of the program's modules

As mentioned before, the developed code is separated in two main modules attending to the two principal visualization modes offered by this program (see figure 3.2). Is the user, according to its visualization needs, who have to select which module is executed.

Both code scripts call a number of developed MATLAB functions that accomplish specific tasks of the data processing and visualization. This chapter section describes the two main modules and the process followed by each of them to achieve the visualization.

4.2.1 Two dimensions visualization

This script is in charge of performing a two dimensions (2D) visualization of a plane, according to a range of available conditions. The script is divided in seven code sections, creating a pipeline from the selection of the data to its visualization, as can be observed the diagram presented in 4.3.

Figure 4.3 – Diagram of the pipeline followed by the 2D visualization script.

In order to be ready to run and deliver the data visualization, the script needs to have the data files in the program folder, and the name of the data source selected to visualize as input for the script. Although the script has a set of default visualization options for each data source, the user can change the options in order to adjust the visualization to his expectations.

The following describes each code section of this module in order to give the reader a deeper knowledge of how the data processing and visualization is performed.

4.2.1.1 Input

The first subdivision of the module is essentially dedicated to the selection of a data source and visualization options. The code receives the data source name from the user saving it into the program internal information structures. Furthermore, this section of the code allows the user to set specific options to customize the process and visualization of the selected data source.

It must be also mentioned that the code is prepared for both already known data sources and new ones, being really easy to introduce the new ones.

4.2.1.2 Default

With the objective of making the program easy to use for the researchers, this section of the code gives basic default values to the options that have not been filled by the user in the previous section.

Moreover, this section of the code is meant to configure the program internally, avoiding the existence of contradictions in the options selected by the user. Thus prevents the appearance of errors in the program execution.

It must be highlighted that the user can modify the default options of the program to adapt it to his or her interests. This can be done accessing to the code of the function that loads the default values of the options and changing the desired values.

4.2.1.3 Load Data

As can be deducted from its name, the task of this section of the code is to load the input data. Both velocity and B-mode data from the selected source are automatically loaded to the script. The data will not be processed and saved into the its internal `data` structure until the following subdivision of this script's code.

4.2.1.4 Prepare Data

The final objective of this section of the module is to fill the program's internal structure `data` with the loaded data, prepared for the visualization. In order to do this, the data needs to be processed to a greater or lesser extent. The data adaptation is done through individual functions developed to fit with the nature of the source of the loaded data. Again, the script is ready to receive a new data source, being needed to implement a MATLAB function for its processing; this can be done following the guidances provided in the program.

One of the main tasks performed in these files is the scan conversion of B-mode raw data. To give a better understanding of the nature of the non-interpolated B-mode data that can be received by the program, figure 4.4 shows the process of the B-mode formation from the RF signals received from the transducer to the B-mode matrix.

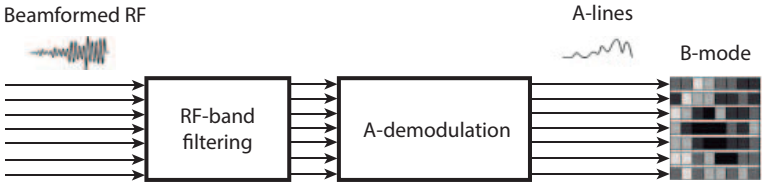


Figure 4.4 – Diagram of the B-mode matrix formation (before interpolation).

It must be pointed out that these A-lines (resulting from the amplitude demodulation of the RF signals), have an amplitude that vary in a high extent. This would make the gray scale image be overshadowed by the appearance of high amplitude points, avoiding the appreciation of important tissue structures. With the aim of overcoming this issue the program gives a high dynamic range to the image matrix, mapping its values to a logarithmic scale. The non-linearity of the logarithmic scale allows this transformation. More information about the B-mode image processing can be found in [3].

Once the B-mode matrix have been built, it needs to be scan converted in order to be displayed, as shown in figure 4.5.

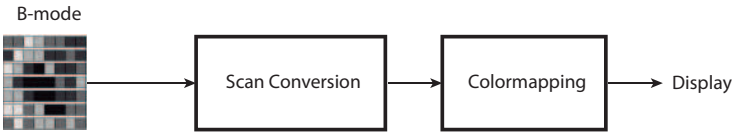


Figure 4.5 – Diagram of the B-mode scan conversion and visualization.

The image corresponding to the B-mode matrix obtained from the A-lines does not maintain the geometry of the source tissue. This is caused by the fact that it does not take into account how the transducer geometry settings affects the data acquisition. Taking into account that the images yielded by this process have clinical purposes, it is crucial that it maintains the geometry of the source tissue. For this reason, the B-mode matrix should be scan converted, which means mapping the image to the actual geometry of the tissue. Figure 4.9 illustrates this process. Using signal processing terminology, this process corresponds to an interpolation.

Figures 4.6 and 4.7 show a B-mode image before the scan conversion and the same image after it, respectively. Figure 4.8 shows the same image with the gray scale that is used in the final visualization.

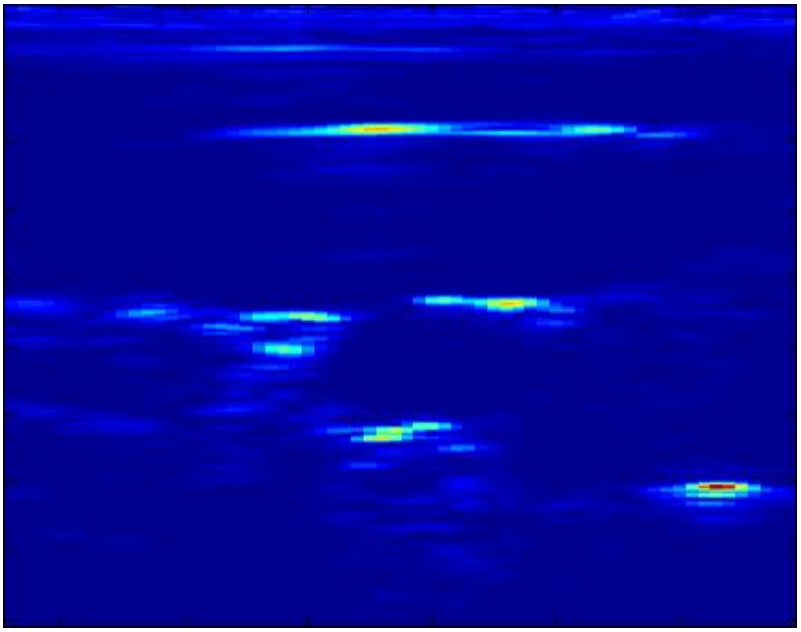


Figure 4.6 – B-mode matrix before the scan conversion. Data provided by Simon Holbek.

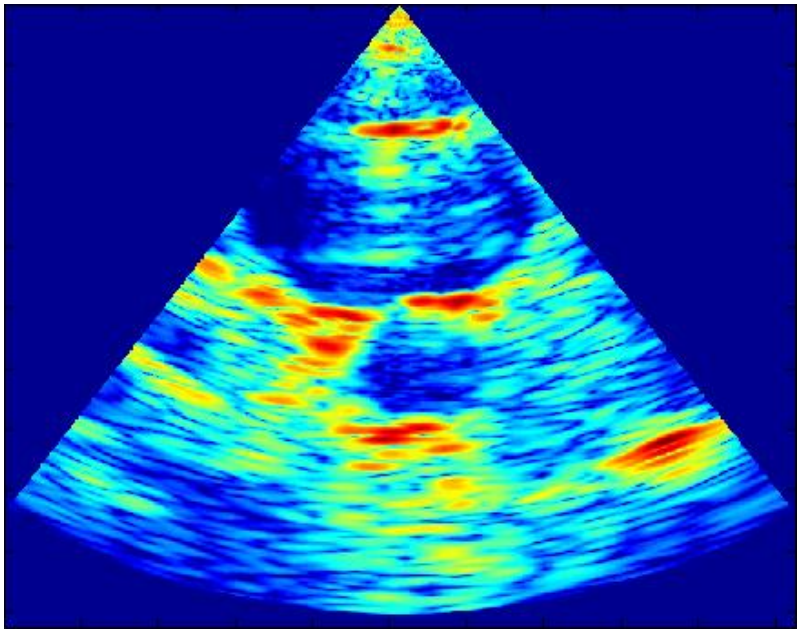


Figure 4.7 – B-mode matrix after the scan conversion. Data provided by Simon Holbek.

Figure 4.8 – B-mode image in gray scale. Data provided by Simon Holbek.

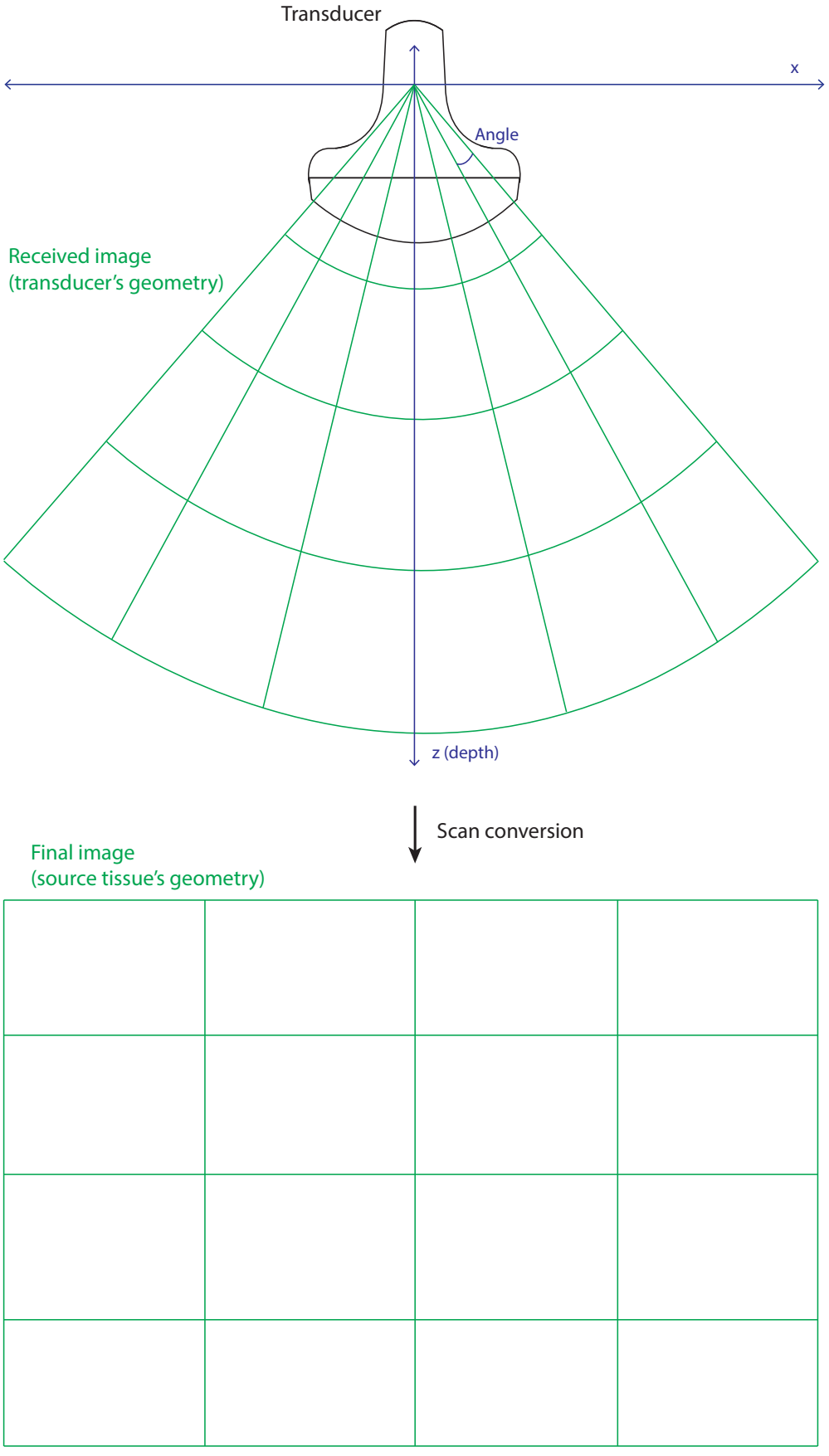


Figure 4.9 – Simplified diagram that illustrates the scan conversion process. This process converts the image from the transducer's geometry to the source tissue's geometry. The transducer has been placed in the upper part of the image, where x, z and the angle outline its geometry.

Beyond the geometry of the measurements, it must be taken into account that some researchers only find relevant certain region of the measured tissue. This should be considered before doing the image scan conversion, in order to choose an appropriate parameters for the interpolation. With the aim of solving the issue, the measurements that cover an extensive region can be processed following the steps presented in figure 4.10. Although the program have a fast and interactive procedure for changing the interpolation's start depth and image size, the remaining parameters of the interpolation should be indicated in the program's processing file of each data source. Among the interpolation parameters that the user can modify the following must be emphasized: angle information, image resolution, number of elements in the axes or scaling of the image.

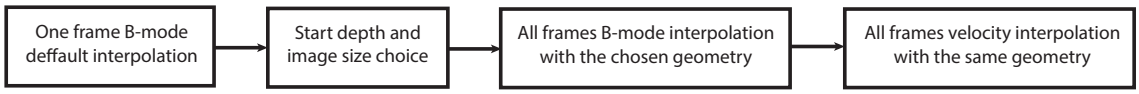


Figure 4.10 – Diagram of the pipeline followed by the 3D data processing.

In this situation, the script interpolates and displays the whole measured region, that would look as the image presented in figure 4.8. The image is depicted using reference axes, with the aim of allowing the user to choose specific values for the interpolation. An example of the B-mode interpolation of an specific region can be observed in figure 4.11. Finally, the velocity values are interpolated to the selected geometry.

As occurs with the static tissue information, the velocity raw data received by the program needs to be processed. In the same way as the B-mode, the raw data from the velocity measurements is affected by the transducer's geometry.

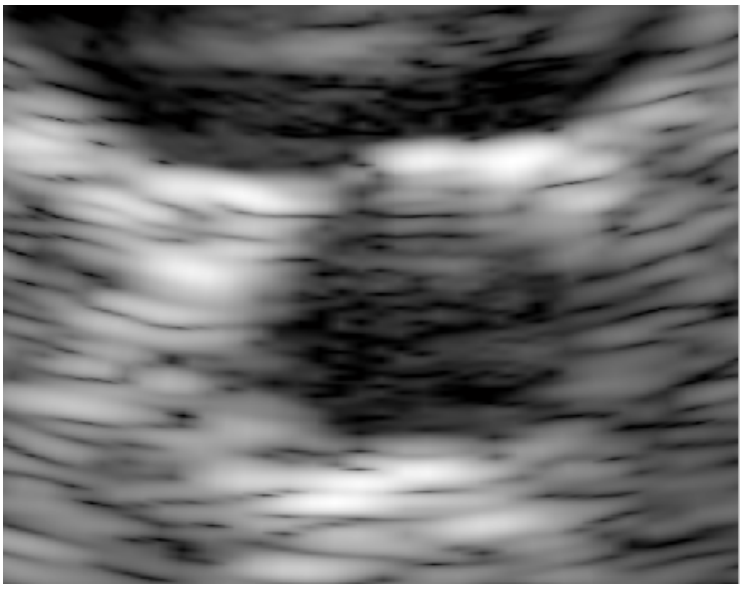


Figure 4.11 – B-mode image interpolated in a certain region. Data provided by Simon Holbek.

Since the processing of the velocity data is performed after the B-mode data processing, the velocity data should always be adapted to the geometry used in the B-mode scan conversion. In such way, the final image will present a realistic representation of the flow and static tissue in the studied region.

The velocity scan conversion should take into account that the velocity measurements' geometry might be different from the B-mode measurements' geometry. It must be highlighted that, so far, the program have not faced any couple of B-mode and velocity data measured using the same transducer's geometry settings.

A simple example of the difference between the geometry of the B-mode and velocity data can be observed in figure 4.12. In this figure, the B-mode shows a vessel that goes from one side of the image to the opposite one, while there is only velocity data for the part of the vessel that appears in the central part of the image. Figure 4.14 shows more clearly the difference between size of the region where there is velocity data available and the size of the final image. It must be remarked, however, that the possible differences between the measurements' geometry are not only limited to the size of the data regions, but also the rest of parameters related to the transducer geometry.

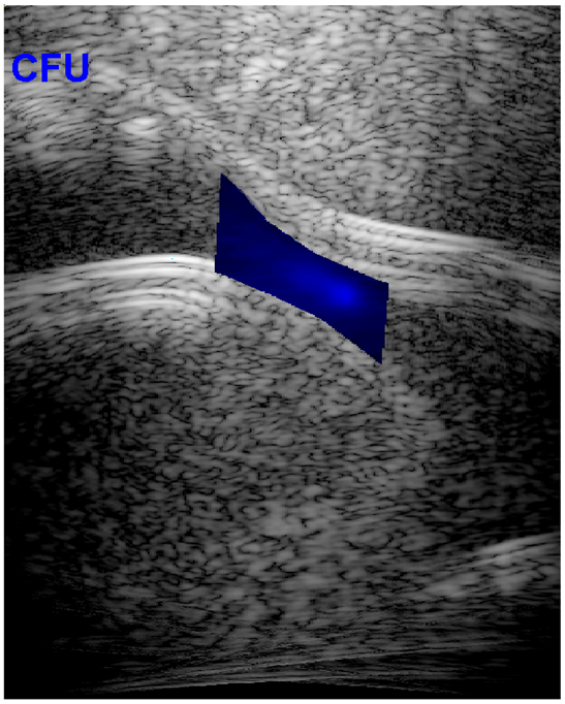


Figure 4.12 – Vessel visualization that illustrates the difference between the sizes of velocity and static data. The blue region of the image represents the velocity data available. The color blue implies that the velocity values in this region are negative. Data provided by Carlos Armando Villagómez Hoyos.

Independently of the different geometry settings, after the corresponding interpolations, both B-mode and velocity data finally fit in equidimensional matrices. These matrices will present a reliable representation of the measured flow and the static tissue that surrounds it.

4.2.1.5 Mask

According to [43], a mask is *"a binary image consisting of zero and non-zero values. If a mask is applied to another [...] image of the same size, all pixels which are zero in the mask are set to zero in the output image. All others remain unchanged"*. Using the same definition, the program uses a 2D mask to combine the B-mode and flow images in the final visualized image. This is done by:

1. Applying a mask of ones and zeros to the velocity data matrix, saving the velocity values that should be shown (the mask have ones where the final image should show the velocity).
2. Inverting the ones and zeros of the mask.
3. Applying the inverted mask of ones and zeros to the B-mode data matrix, saving the B-mode values that should be shown (the mask have ones where the final image should show the B-mode).
4. Combining the images.

Although this process is performed later in the script, the mask required for it is obtained and saved in this code section. The program offers three different alternatives to obtain the mask:

- Use an already created mask provided with the data.
- Loading data saved in previous interactions of the program with the source data.
- Creating a new mask.

In the third case, the user has the option of saving the created mask, in order to use it in forthcoming visualizations of the same data.

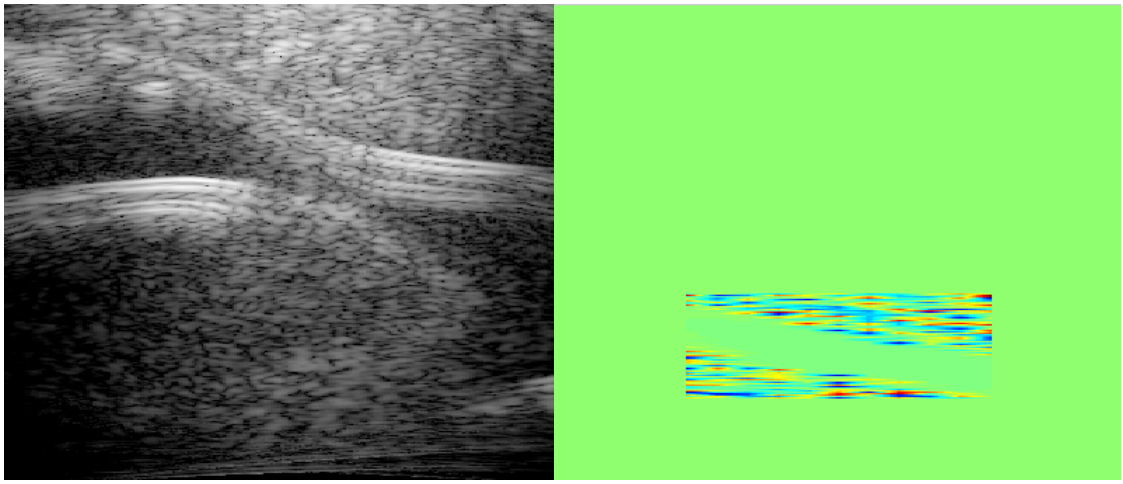
The mask creation can be done automatically when having velocity data with appreciable difference between the static tissue points and the ones with flow. However, since the program users are researchers working in velocity estimation, the program needs to be ready to velocity files with irregularities. For that reason, users also have the possibility of creating the mask delimiting manually a region with one or more polynomials. This mask creation process consists in the following:

1. The user can choose between using the B-mode or the flow matrix as a model to draw the mask on top of it.
2. The model image chosen is plotted.
3. The program requests the user to delimit a region or more than one regions (depending on the chosen options) in order to create the mask from the image. The request is done through a message box like the illustrated in figure 4.13.
4. Once the mask is delimited, the user can modify it until the result fulfils the derived requirements.
5. When the user indicates that the mask is ready it is saved.

(a) One polynomial. **(b)** More than one polynomials (four in this case).

Figure 4.13 – Message boxes requesting one or more than one polynomials to create the mask.

Figure 4.14 shows both a B-mode image and a Velocity image that act as models for the creation of the dataset mask. Both figures correspond to the same dataset, although their geometry is not the same. This issue is solved when building the final image to visualize, taking into account the measurements' geometry information provided with the datasets in SARUS measurements. Figure 4.15 shows a polynomial function delimiting the velocity region that will appear in the final image, with the aim of building the mask. It should be also mentioned that the program offers the possibility of creating a mask for the whole dataset or creating one for each frame.



(a) B-mode image model for the mask. **(b)** Velocity image model for the mask.

Figure 4.14 – Image models for the mask creation. Data provided by Carlos Armando Villagómez Hoyos.

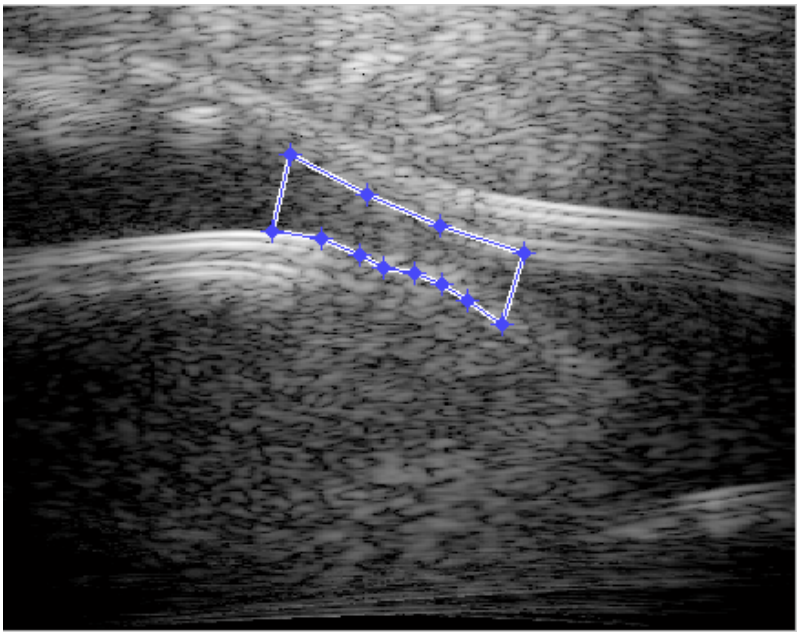


Figure 4.15 – Polynomial function delimiting the mask. The region delimited by the blue lines will display the velocity data in the final image. Data provided by Carlos Armando Villagómez Hoyos.

4.2.1.6 Visualization 2D

The final section of this module's code completes the 2D visualization process, executing the visualization itself. Since the program offers a big range of different visualization options all of them were taken into account when developing this final subdivision of the code of this 2D module.

There are a wide number of principal visualization options, however, it must be pointed out that most of these options can be combined, offering a wider range of visualization possibilities. The following enumerates the different visualization options offered by the program for the 2D visualization:

- Visualizing the velocity data with a colormap. As can be observed in figure 4.16.
- Visualizing the velocity data with vectors on top of the colormap. As can be observed in figure 4.17.
- Visualizing the velocity data with streamlines on top of the colormap. As can be observed in figure 4.18.
- Visualizing the researching group logo on the corner of the visualization. As can be observed in the figures presented after the enumeration.
- Visualizing either the absolute velocity or any component of the velocity vectors in the region. The following images can be taken as examples: figure 4.16 presents the visualization the component v_z of the velocity, meanwhile figures 4.17 and 4.18 present the visualization of the absolute value of the velocity data.
- Visualizing the velocity data using a certain velocity range (introduced by the user) for the velocity colormap in all the frames.
- Visualizing the velocity data using a certain velocity range (calculated by the program) for the velocity colormap in all the frames.
- Visualizing the velocity data using a defined value (introduced by the user) as maximum for the vectors in the vectors' visualization for all the frames.
- Visualizing the velocity data using a defined value (calculated by the program) as maximum for the vectors in the vectors' visualization for all the frames.
- Visualizing the image choosing the velocity units between: m/s, cm/s and mm/s. As can be observed in the figures presented after the enumeration.
- Visualizing the image choosing the axes' units between: m, cm and mm. As can be observed in the figures presented after the enumeration.
- Visualizing the image choosing the number of labels that appear in the colorbar.
- Visualizing the image choosing the number digits that appear in the colorbar labels.
- Visualizing the velocity vectors determining the size of the matrix that contains its origins.
- Visualizing the velocity streamlines determining the size of the matrix that determines where they are depicted.
- Visualizing the velocity streamlines being able to choose their starting point.

- Displaying an auxiliary graph next to the velocity visualization, showing the velocity against time on a certain point of the image. As can be observed in figure 4.19.
- Displaying an auxiliary graph next to the velocity visualization, showing the velocity values of a certain segment of the image. As can be observed in figure 4.20.
- Visualizing the data including the time of each frame in the visualization.
- Visualizing the data choosing the number of different B-mode values in the gray scale.
- Visualizing the data choosing the number of different velocity values in the velocity color scale.
- Visualizing a dataset with one B-mode data for several frames of velocity data.
- Visualizing one specific frame of a multi-frame dataset.
- Generating a *gif* image from the set of time frames of the dataset, as they correspond to the measurement of data in a time lapse with certain sampling period.
- Generating a video from the set of time frames of the dataset, as they correspond to the measurement of data in a time lapse with certain sampling period.
- Generating the video or *gif* image with a certain time frame between the frames.
- Generating the video or *gif* image without visualizing each frame in the process.

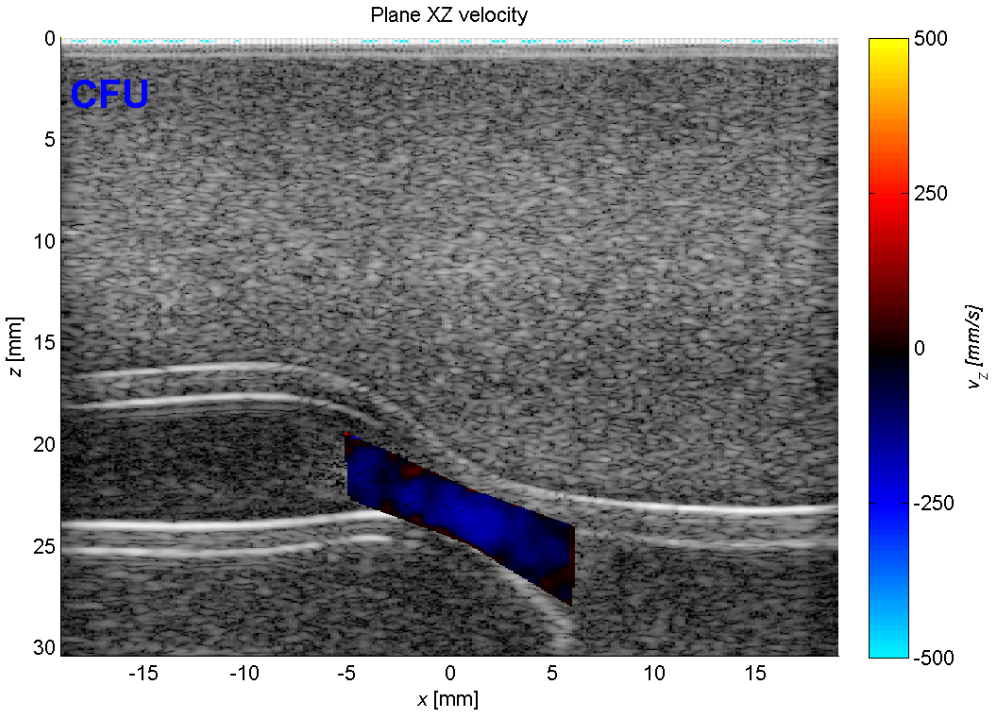


Figure 4.16 – Basic visualization of the velocity data (colormap). The warm colors depict positive velocity values, while the cold ones depict negative velocity values. This visualization shows the z component of the velocity. Data provided by Jacob Bjerring Olesen.

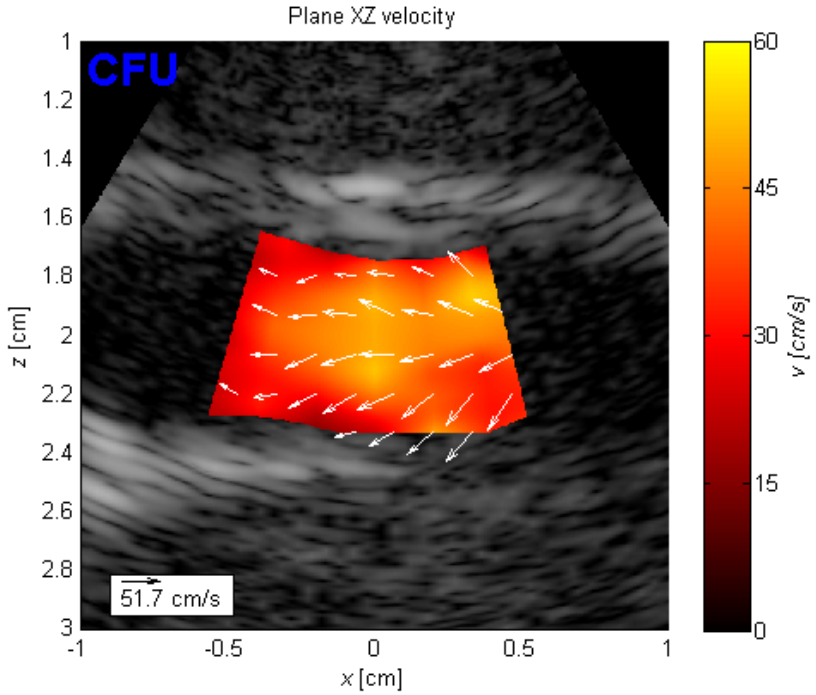


Figure 4.17 – Velocity data visualized through vectors and colormap. The colormap visualization shows the value of velocity in each point of the region. The vectors depict the two in-plane components of velocity (v_x and v_z). The vectors' legend is located at the bottom left. Data provided by Simon Holbek.

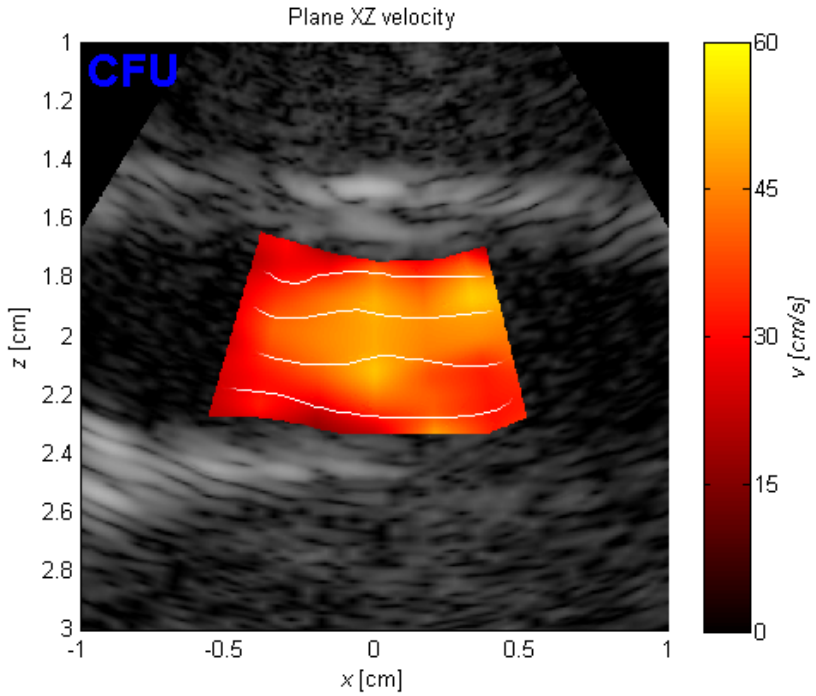


Figure 4.18 – Velocity data visualized through streamlines and colormap. The colormap visualization shows the value of velocity in each point of the region. The streamlines depict the behaviour of the two in-plane components of velocity (v_x and v_z). Data provided by Simon Holbek.

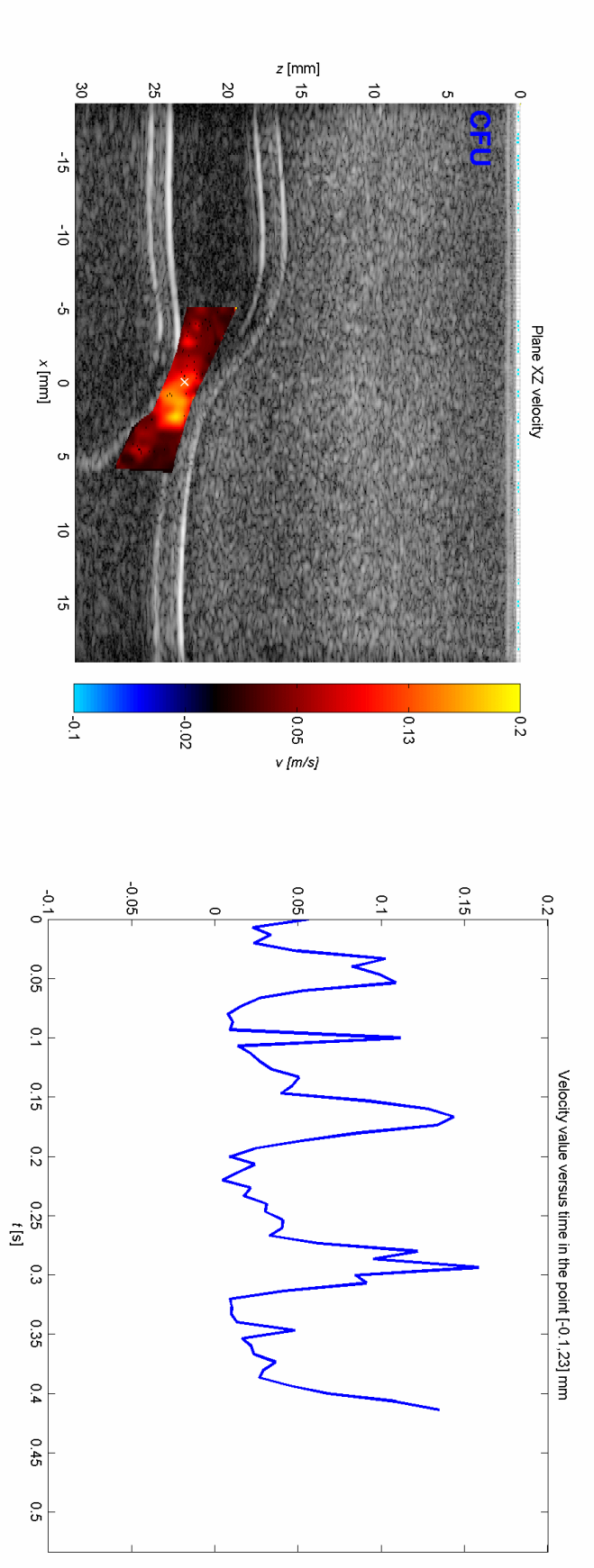


Figure 4.19 – Visualization of the velocity in one point against time, using an auxiliary plot. The chosen point has been plotted through a white "x". The velocity values in this point shows the pumping behaviour of blood. Data provided by Jacob Bjerring Olesen.

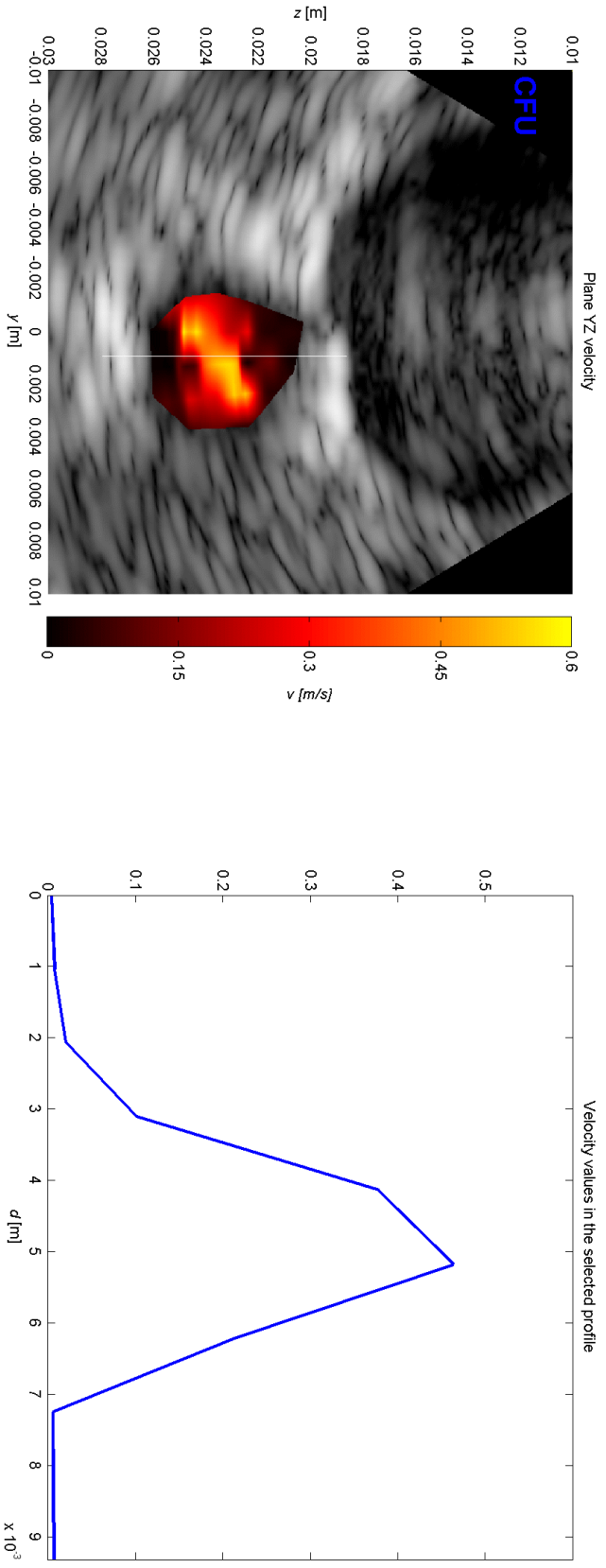


Figure 4.20 – Visualization of the velocity values in one segment of the image, using an auxiliary plot. The chosen segment has been plotted in white. The velocity profile shows the laminar flow of blood. Data provided by Simon Holbek.

Before performing the visualization itself, the code checks if the user has selected one of the two complementary plots available. Either a plot of the velocity values in a segment of the image selected by the user, or a plot of the velocity values in one point of the image against time. After having checked this, if any of them are enabled, the program requests the user to select either the segment or the point, related to the mentioned complementary plots.

As next step, if the user has selected a time changing visualization, whether it is a *gif* image or a video, the program creates a folder in order to save together all the visualization frames in an orderly way. After this, depending on the option selected by the user:

- The program visualizes a unique frame of the dataset.
- The program visualizes all the frames of the dataset. The user must be aware of the fact that this option is susceptible to give problems when dealing with big datasets with a high number of frames if the computer's RAM cannot handle the memory required.
- The program prints as images the visualization of all the frames of the dataset (saving them in the created folder).

When choosing the last option, in order to save the images, all the frames are plotted as invisible figures, in order to prevent problems with the RAM memory of the computer having to manage a lot of simultaneous MATLAB plots at the same time or the creation and closing of a figure in each interaction. By using an invisible figure the script improves its speed.

2D image creation

The visualization of each data frame is performed by a function. The code of this function performs the 2D visualization itself, visualizing a matrix with the velocity data on top of the corresponding B-mode. The following lines detail the image building and visualization performed by this function.

The code, first, chooses the default number of decimal digits for the colorbar labels depending on the units requested for the velocity representation. When determining the default value for this number of digits, it has been taken into account that the peak velocities in the main vessels of the human body go from a few dozen to a few hundred centimetres per second [2]. Then, in order to perform the requested visualization, the function checks the plane to visualize, and prepares the data for its visualization loading it in variables and adjusting the units if it is needed. The velocity that will be plotted through the colormap is selected.

As the data comes in two different matrices, B-mode and velocity, they need to be combined in order to build the final image. With this purpose, an inverse mask is generated from the given mask, changing the ones with zeros and the zeros with ones. From now on, the given mask will be called velocity mask and the new one B-mode mask, making reference to the regions that should be visualized where the mask have ones. Both masks are used later to separate the relevant points of each matrix and combine them.

As next step, the B-mode data is normalized, taking into account that it comes in a logarithmic compressed range between -60 and 0. Then, it is multiplied by the number of B-mode values selected for the image building. When choosing this value, the number of shades of gray that an human observer is able to difference becomes relevant. According to [24], humans are able to discriminate *"between 700 and 900 simultaneous*

shades of gray" in medical displays. However, a balance should be found between the capacity of the image for being representative and the executing time that takes to yield it. Also, the range of values obtained from the measurement and the importance of the detail of the B-mode representation should be taken into account. The experience has shown that an convenient value for the number of B-mode values in the program representation is 127. For that reason this value has become the default value for this parameter. However, the users can change this value easily through the program's options.

Finally, the matrix is multiplied by the mask, obtaining a B-mode matrix with zero in the positions where the velocity will be depicted.

The velocity data preparation for the visualization is slightly more complex due to its variable range of values. Thus, the velocity selected to be visualized through the colormap will be normalized depending on the options selected by the user. There are three different ways in which the velocity range can be obtained for visualizing the velocity matrix:

- Velocity range selected by the user through the program's options.
- Velocity range calculated by the program for the whole dataset, saving the maximum and minimum of all the dataset frames.
- Velocity range calculated by the program for each frame. Default option.

These three different choices for the velocity range acquisition affect the matrix normalization. For the first and second cases, the matrix will be normalized with the range values, which are provided to the function. In the third case, the code has to calculate the range for the frame. The function first obtains the maximum and minimum values of velocity in the region that will be visualized using the mask to evaluate the matrix only in the interesting points. Taking into account that these maximum and minimum values would not be exact numbers or numbers with a few decimal digits, rounding is applied to each one: rounding up to the maximum value and rounding down to the minimum one, in order to keep all the velocity values inside this range. Then, the velocity matrix is normalized.

To perform the normalization, the velocity matrix without the mask is used (because the zeros in the matrix also would be normalized if the matrix with the mask applied is taken). It must be taken into account that the velocity values being normalized can be positive or negative and after the normalization all the values will be compressed between zero and one. If the user has selected a velocities range, the matrix values that are out of the range are changed to the range limit values. Then, the normalization is carried out with the range values. After that, in both cases, having the velocity matrix normalized, it is multiplied by the number of velocity values selected by the user. This value can be chosen by the user through the program's options. Again, the default value for this parameter is 127, as it has proven to provide a fast visualization with a sufficient level of image quality.

Finally, the matrix is multiplied with the mask, obtaining a velocity matrix with zeros in the positions that correspond to the B-mode.

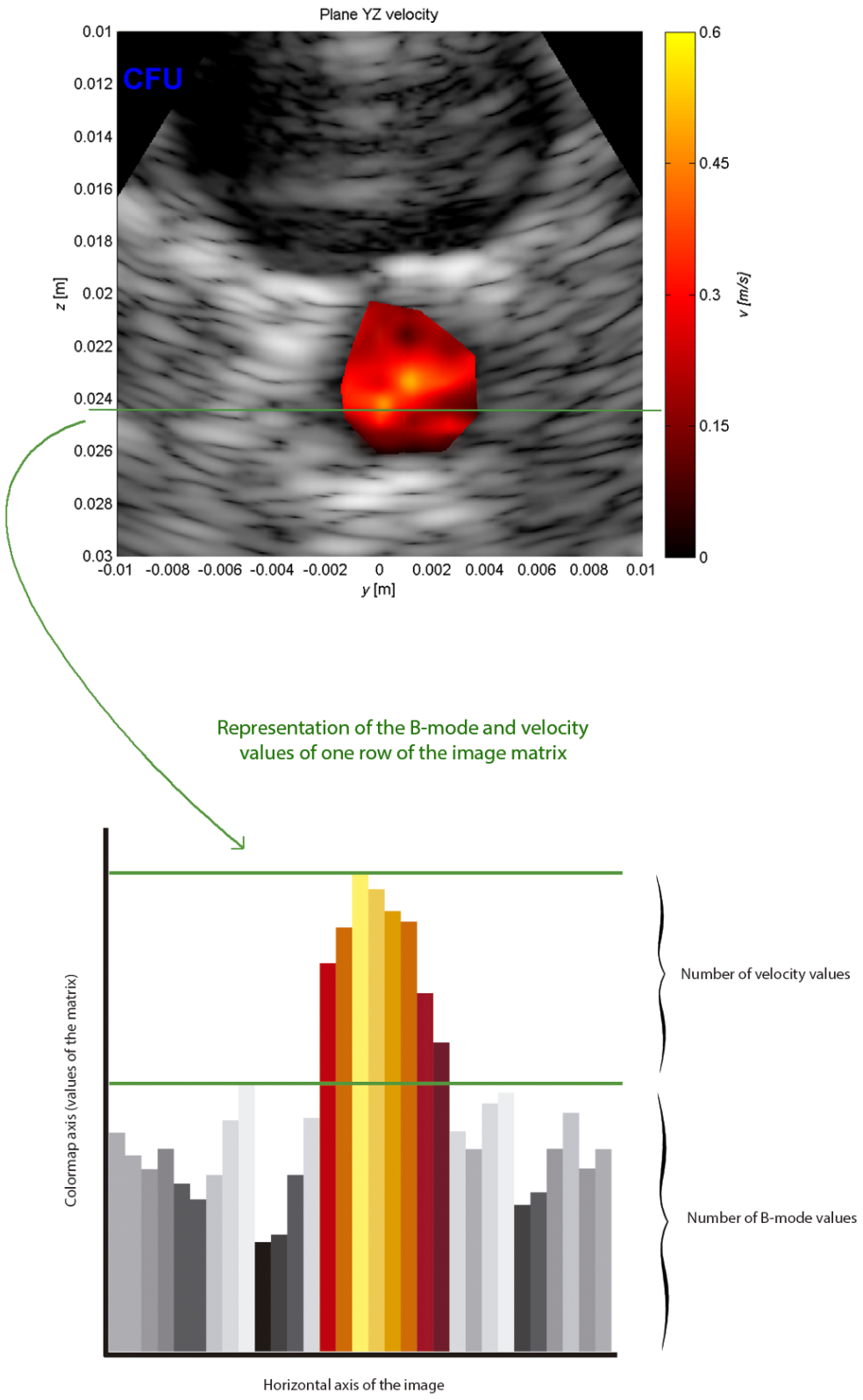


Figure 4.21 – Representation of the values of a matrix's row from the B-mode plus velocity image matrix. The values of the row have been depicted in a graph that shows the B-mode values in gray colors and the Velocity values in the colorbar's color scale (warm colors in this case). The vertical axis of this graph corresponds to the image's colormap, meanwhile the horizontal axis corresponds to the horizontal axis of the image ($y[m]$). Data provided by Simon Holbek.

After having prepared the B-mode and velocity data the result is a pair of matrices with values from zero to the number of values selected for each one. As has been said, by default, both matrices can have 127 different values for each element of the matrix. These two matrices are combined by:

1. First an offset is applied to the velocity matrix in order to have all its values above the number of B-mode values.
2. Then both matrices are combined. B-mode values will be in a range from zero to to the number of B-mode values and velocity values in a range from the number of B-mode values to this value plus the number of velocity values.

A diagram that clarifies the structure of the final image's matrix can be observed in figure 4.21. The figure presents a visualization obtained with the program and a representation of the values of one of the rows of the image matrix. B-mode points (gray scale) have values under the number of B-mode values, while the velocity values (warm scale) are above it. It should be mentioned that, if the velocity is zero in the whole given region, the code reduces the number of velocity values to one, in order to economize resources and favour a fast visualization.

Once the matrix has been created, and before plotting it, vectors or streamlines are generated if the user requests it, in order to plot them on top of the matrix visualization.

The vectors' generation starts with the creation of a matrix of positions with less rows and columns than the image (the user can choose the size of this matrix). The reason behind the smaller size of the matrix is that this matrix will contain the origin of the vectors, and if it is too dense the vectors would overlap in the image, being impossible to perceive any illustrative velocity information. Having a vectors' origins matrix with less columns and rows than the main matrix would allow to plot the vectors with some free space between each other. Then, the velocity mask is applied to both vector components (to eliminate the vectors outside the velocity region), and the matrix of each vector component is sampled to take the vectors corresponding to the new matrix of positions.

Although MATLAB's function `quiver` (that visualizes arrows) resizes the vectors itself for its representations, the possible proximity of vectors in this kind of visualizations causes the possibility of a messy overlapped visualization and, moreover, the management of the resizing would be out of the program control. For these reasons, the vectors are normalized and resized by the developed code, avoiding the overlap among adjacent vectors. Depending on the user's choices, the vectors' normalization is performed with one of the following options:

- Maximum velocity vector value selected by the user through the program's options.
- Maximum velocity vector value calculated by the program for the whole dataset, saving it from all the dataset frames.
- Maximum velocity vector value calculated by the program for each frame. Default option.

After the normalization, the vector's components matrices are resized and prepared for the visualization.

The streamlines follow the same process as the vectors for their preparation due to the similarities between their correspondent MATLAB's functions. Attending to the definition given by [44], in fluid mechanics, a

streamline is *"the path of imaginary particles suspended in the fluid and carried along with it"*. However, the streamlines visualized by the program's code show the velocity values in a certain time, not corresponding to the values of the same particle across time. The program streamlines have been developed in such way due to the value that they add for some specific cases, when they meet the researchers needs. On the other hand, based in experience, it can be said that the most used options to present the velocity values are the other two: the colormap alone or the colormap with vectors. Lastly, it must be mentioned that it is not excluded that streamlines that concord with the definition can be implemented in the future, if this means the increase of the program's value for the researchers.

After having dealt with the streamlines or vectors (if it was needed), the visualization is ready to be yield depending on the options selected. The previously built B-mode/velocity matrix is depicted and, if it has been requested, vectors of streamlines are plotted on top of it.

Once the image is visualized, the axes, the colormap and the colorbar should be managed. Addressing first the handling of the colormap and the colorbar, it should be mentioned that the construction of the colormap is not an simple task, since its creation process changes depending on the velocities depicted.

When designing the program, it was decided to depict the values corresponding to the B-mode in a gray scale and the velocity with a color scale that fulfils these premises: positive values of velocity are depicted in a warm colors gradient, meanwhile the negative values are depicted in a cold colors gradient. Moreover, in each gradient, the dark colors correspond to a low value of velocity (positive or negative) and the light colors represent high velocity values (positive or negative).

The building of the four different possible colormaps is carried out by, first, evaluating the maximum and minimum values of velocity, making the colormap velocity range a bit bigger (rounding up or down for the maximum and minimum respectively) if it improves the appearance of the colorbar. It must be remembered that the users also have the possibility of choosing the velocity colormap range of their liking. The following presents the four different comormaps that can be generated by the program, depending on the data velocity values:

- Velocity equal to zero: A black colormap is used to represent the lack of velocity. An example of this colormap can be observed in figure 4.22.
- Positive velocity values: The developed warm colormap (hot_flow) is applied. An example of this colormap can be observed in figure 4.23.
- Negative velocity values: The developed cold colormap (cold_flow) is applied. An example of this colormap can be observed in figure 4.24.
- Positive and negative velocity values: The value of zero velocity is found and the warm colormap (hot_flow) is assigned to the positive values, while the cold one (cold_flow) is assigned to the negative ones. An example of this colormap can be observed in figure 4.25.

In all the cases the colorbar only shows the velocity values of the colormap, being understood that the goal of the program is to offer an illustrative visualization of the flow. If the group researchers needed it the colorbar could have been programmed to show the B-mode values as well.

The implementation of the management of the colorbar and the colormap required several considerations, since the image values are different depending on the data, but also because of the processing followed for the image creation (normalization of the matrices and combination of them).

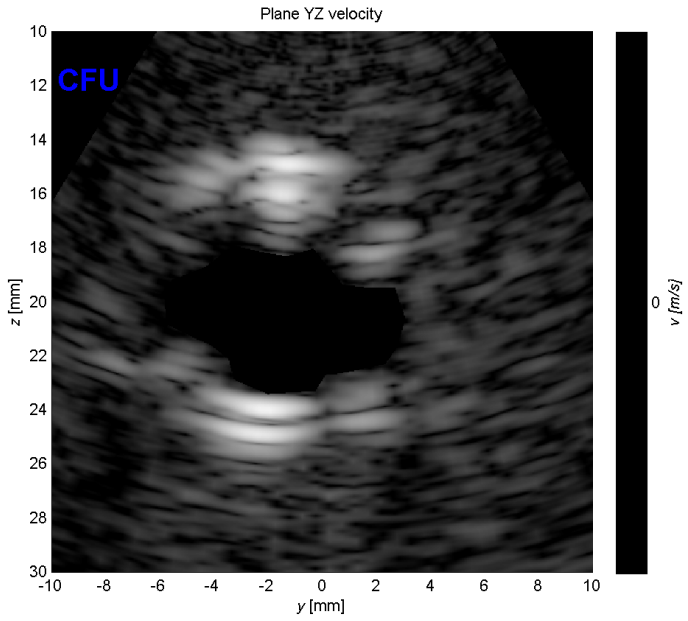


Figure 4.22 – Velocity visualization of static data. The colormap visualization shows the value of velocity in the region: zero. Modification of a dataset provided by Simon Holbek.

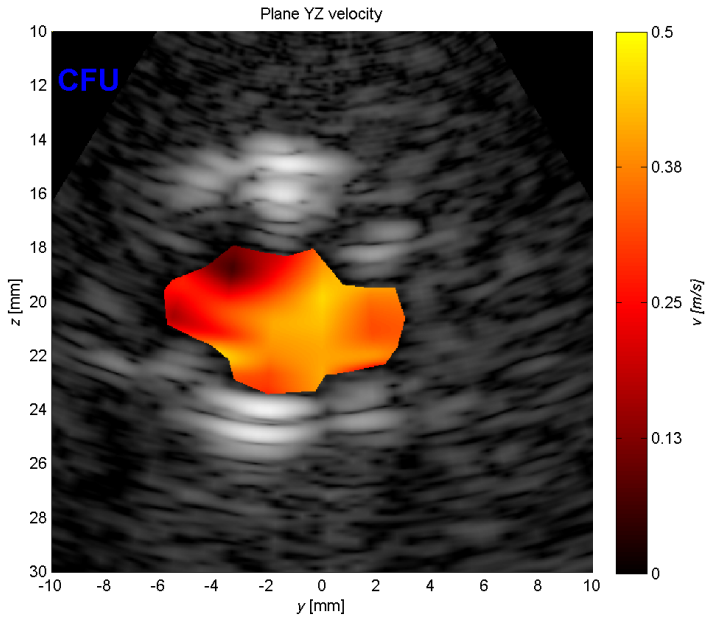


Figure 4.23 – Velocity visualization of flow data with only positive values. The colormap visualization shows the value of velocity in each point of the region. Modification of a dataset provided by Simon Holbek.

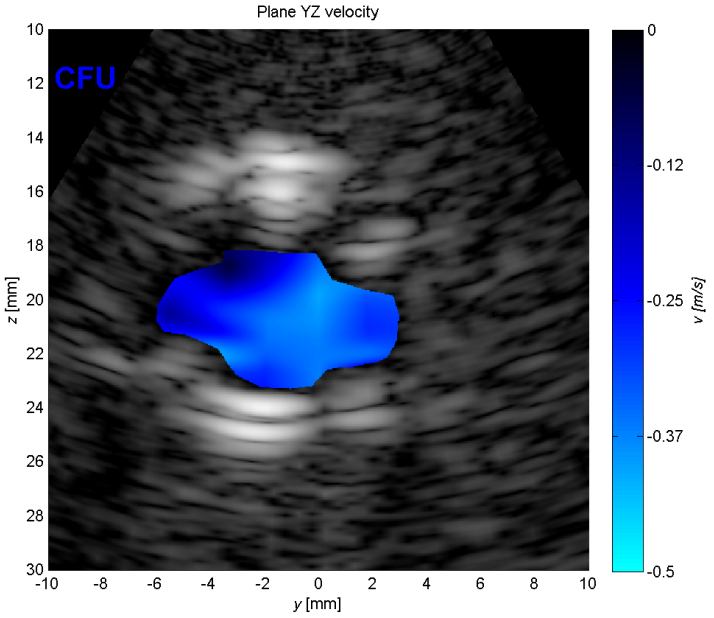


Figure 4.24 – Velocity visualization of flow data with only negative values. The colormap visualization shows the value of velocity in each point of the region. Modification of a dataset provided by Simon Holbek.

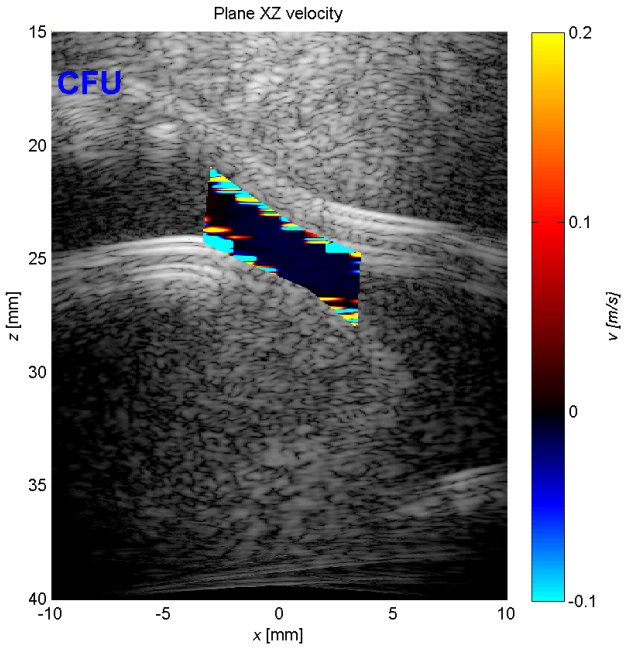


Figure 4.25 – Velocity visualization of flow data with both positive and negative values. The high positive and negative values that appear in the velocity data bring to light an error in the researcher's velocity estimation algorithm for values near zero. Data provided by Carlos Armando Villagómez Hoyos.

The colorbar labels and values are set according to either the options chosen by the user or the default ones. The title and axes' labels are generated taking into account the characteristics of the requested visualization.

Then, the research group's logo is plotted on top of the image and a legend with a reference vector is depicted in case of the vectors visualization (with the aim of providing a reference with which the user can relate the size of the data velocity vectors).

If the user has chosen any auxiliary plot for the visualization, either a point or a segment is plotted on top of the image, in order to complement the information shown in the auxiliary plots: velocity against time in a point or velocity profile in a segment of the image.

Having described the 2D image creation function, the following paragraphs continue explaining the code of the main 2D visualization script. After calling the image creation function several options for the visualization should still be considered, like the *gif* and video creation or the auxiliary plots.

Once the main visualization of the data is performed, if the user has selected the related options, the selected complementary plot is visualized for all the frames and saved in the created folder. A visualization example with each auxiliary plot can be found in figures 4.19 and 4.20. The following lines describe the code that creates these two functions.

Velocity against time auxiliary plot

This function plots the velocity value of a image point (selected by the user) against time. The function receives the selected point and then generates the auxiliary plot for all the dataset frames.

In order to do so, the function needs to select the appropriate range of values for the plot's y-axis. If the user has selected a range for the velocities, the velocity data matrix may need to be truncated. Once the range for the plot's y-axis has been defined, the function goes through all the frames creating the plot of velocity against time of each frame by adding the new value to the one from the previous frame. All the frames are plotted as invisible figures, with the mentioned advantages this gives to the visualization. The frames are saved into the corresponding folder so the function has no output (beyond the error control). An example of the velocity graphs generated and saved by this function can be observed in figure 4.26.

Velocity profile auxiliary plot

This function plots the velocity profile in the image segment selected by the user. The function receives the selected segment and then generates the auxiliary plot for all the dataset frames.

To achieve this task, the function selects the appropriate range of values for the plot's y-axis, following a process similar to the described for the previous function. In this case, however, a segment must be created from the two saved points. A 100-points segment is created with the size of the distance between the two saved points. The two values of the coordinates of all the line points are calculated. Then a 2D interpolation is used to obtain the velocity value for each point of the segment in each frame. This interpolation is required to overcome the possibility of needing a value of velocity that is not corresponding exactly to any cell of the velocity matrix. Finally the function runs over all the data frames creating the velocity profile plot in the segment. All the frames are plotted as invisible figures and saved into a folder, not having any function output (beyond the error control), as occurred with the other auxiliary plot function. An example of the velocity graphs generated and saved by this function can be observed in figure 4.27.

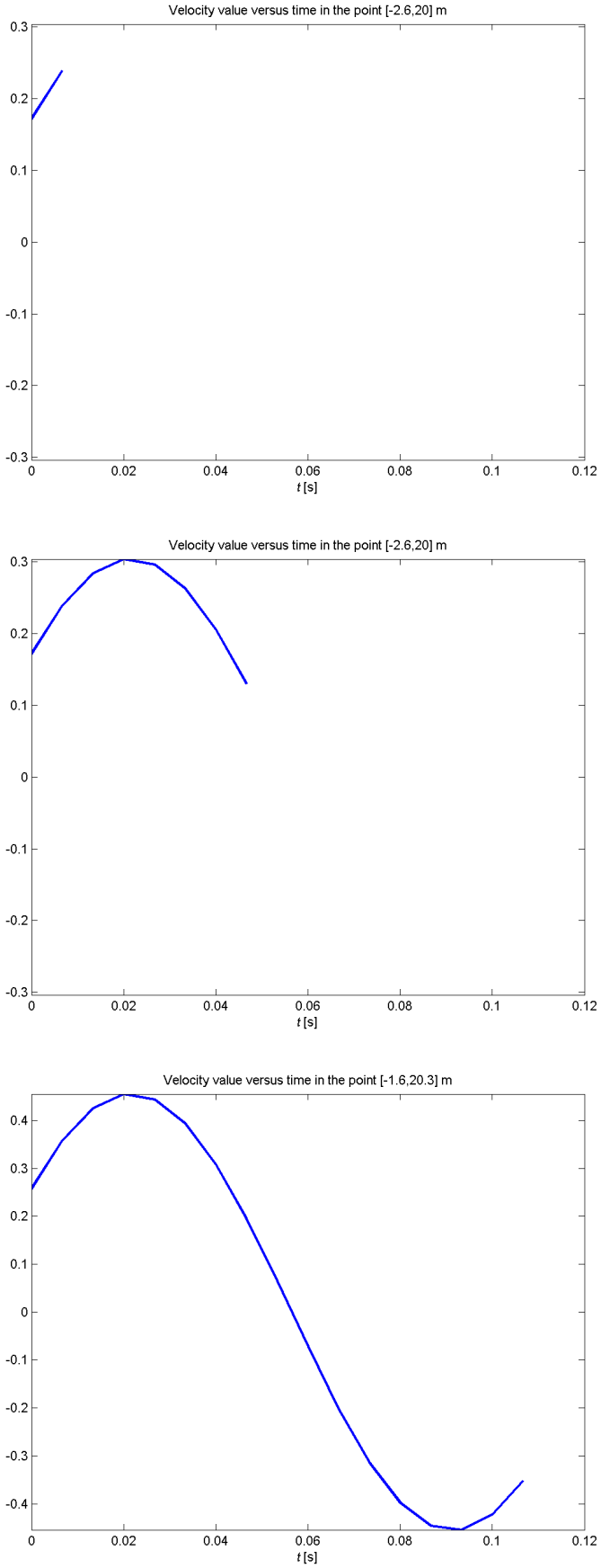


Figure 4.26 – Velocity against time auxiliary plot in three different frames of the same dataset. The images have been ordered according to their frame number, from lower to higher. The dataset proceed from measurements of flow from a controlled flow-rig system, which explains its sinusoidal behaviour. Data provided by Michael Johannes Pihl.

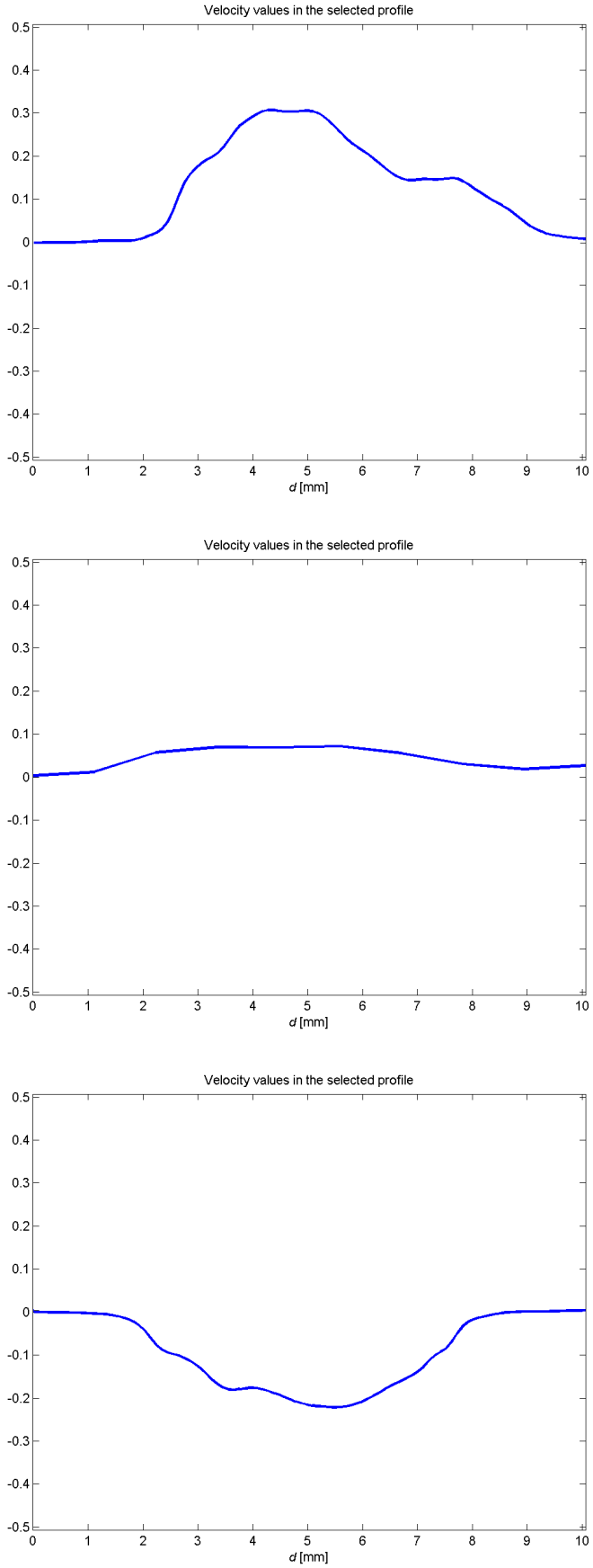


Figure 4.27 – Velocity profile auxiliary plot in three different frames of the same dataset. The images have been ordered according to their frame number, from lower to higher. These three images evidence the pulsating behaviour of the blood flow. Data provided by Simon Holbek.

Gif or video creation

Finally, in the main 2D visualization script, the *gif* or video is created if the user has requested it.

To create the *gif*, the code difference between the regular visualization and the visualization with an auxiliary plot: in the second case, the the script combine the basic image with the auxiliary plot of each frame before adding the frame to the *gif*. The *gif* is created through a loop, creating first the *gif* file and adding the subsequent frames in the following iterations.

In the case of a video generation having been requested, the code again differences between the regular visualization and the visualization with an auxiliary plot, combining the plots if needed. The video is created using modifications of the researching group's video functions. These complex functions only operate in Unix operative systems. The development of the homonymous functions for other operative system can be seen as future work for improving the developed program.

4.2.1.7 Calculate velocity range and/or vectors maximum

This subdivision of the code is actually placed before already presented in 4.2.1.6, Visualization 2D. The reason behind changing the order of describing them in the report is because this code section performs specific tasks for certain visualization options that would only be understood after comprehending the visualization process.

As it has been already mentioned, there are three choices regarding the acquisition of the velocity range for the velocity matrix normalization and visualization:

- Velocity range selected by the user through the program's options.
- Velocity range calculated by the program for the whole dataset, saving the maximum and minimum of all the dataset frames.
- Velocity range calculated by the program for each frame. Default option.

Also, there are three choices regarding to the reception of the maximum value for the velocity vectors plotted on top of the colormap:

- Maximum velocity vector value selected by the user through the program's options. This option should be requested by the user.
- Maximum velocity vector value calculated by the program for the whole dataset, saving it from all the dataset frames. This option should be requested by the user.
- Maximum velocity vector value calculated by the program for each frame. Default option.

This subdivision of the code is only relevant when the user requests the program to calculate the velocity range or maximum vectors value for the whole dataset.

Calling a function for this task, it calculates the velocity values range or/and maximum for the velocity vectors through all the velocity frames. The procedure used to achieve this is parallel to the calculation of

the same values performed for each frame, however, this function does it for all the data frames, saving the absolute range or maximum values at the end of each loop interaction.

It can be a good decision for the program users to employ this subdivision of the script to calculate the range and/or maximum in the first contact of the program with a new data source. If the same data is visualized more than once, the user can use the calculated values (or choose appropriate values knowing the range and maximum) to visualize the data in the subsequent utilizations of the program. Doing this would save time and resources in the subsequent visualizations of the same data.

4.2.2 Three dimensions visualization

The second main script has the task of performing a three dimension (3D) visualization of two given cross planes of data, and the 3D vectors of velocity of each of them. Being XZ and YZ the two cross planes visualized, a representation illustrating those planes in a MATLAB's 3D space can be observed in figure 4.28.

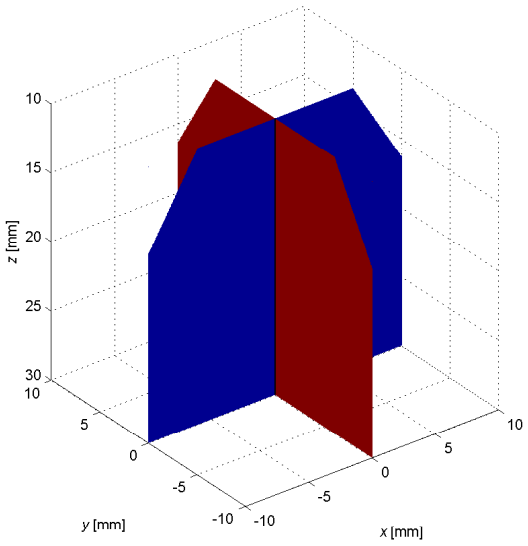


Figure 4.28 – Two cross planes visualized in a 3D space.

The visualization is done according to the conditions selected by the user. The code of this script is divided in seven code sections, whose order can be observed in the diagram depicted in figure 4.29. The pipeline followed by the two main scripts (2D and 3D) is similar, with the intention of facilitating the users' utilisation of the program.

However, the output is not the same as in the 2D script. It seems logical to state this, since the output of a 3D visualization script is expected to be different from a 2D one, but the difference goes beyond the number of dimensions of the visualization environment. The 3D script has been implemented to operate with only one frame of the dataset, as it fits different needs of the researchers. This script, thus, offers the possibility of displaying a different motion visualization than the previous one, which is focused on the time changing

characteristic of the datasets. The motion visualizations (*gif* image and video) offered by the 3D script give a complete view of the 3D visualization by turning the view around the two cross planes. If it becomes a demand, the implementation of the time-changing motion visualization for datasets with more than one frame can be performed with a code similar to the used in the 2D script.

Figure 4.29 – Diagram of the pipeline followed in the 3D visualization script.

Previously in this report (section 3.3.1), the researching US scanner SARUS was described through the following quote from 2013: *"the first of its kind in the world and the only instrument yet created that can display the movement of blood in 3D"* [34]. Taking this into account, it appears logical to find that this script of the program has only received data from one source: SARUS.

The code for accepting more than one data source can be easily implemented in this script following the design that accepts several different data sources in the other main visualization script of the program (2D visualization). Both main visualization scripts, 2D and 3D, have a similar organization of the code in subdivisions, to facilitate the users labour.

The script needs to have the data files in the program folder, in order to run and deliver the data visualization. It is also needed that the name of the data source is given as input for the script. As with the 2D visualization main script, the user can change the options in order to adjust the visualization to his expectations, although the script has a set of default visualization options already selected.

Each section of this module's code is described in the following subsections, giving the reader a deeper knowledge of the process followed by the data until its 3D visualization.

4.2.2.1 Input

As occurred with the 2D main script (4.2.1.1), this first code's subdivision is dedicated to the selection of the data that is intended to be visualized. Together with this, this section of the code allows the user to set specific options to customize the process and visualization of the selected data source.

As mentioned before, the 3D script is ready to receive data from multiple sources, even though it has only worked with data measured with SARUS using the 3D Transverse Oscillation approach.

4.2.2.2 Default

Following the same steps as the *Default* section in the 2D script (4.2.1.2), this section of the code sets with basic default values the options that have not been filled by the user in the previous section. As occurred with the previous script, this is done with the objective of facilitating the utilization of the program.

This section of the code is also responsible of configuring the program internally, avoiding the existence of contradictions in the options selected by the user. With this verifications the code will prevent the appearance of errors in the program execution. On top of that, if the user has not selected any frame to visualize in a multi-frame dataset, the program asks the user to specify the frame to visualize.

As happened with the 2D script default code section, the user can modify the default options of the program to adapt it to his interests.

4.2.2.3 Load Data

With the purpose of getting the input data, this section of the code loads the data from the selected source. The data format from SARUS used by the program during its implementation is divided in two files for each frame: one corresponding to the B-mode and another for the velocity. An example of the nature of a one-frame velocity data file received from SARUS can be observed in table 3.1. The data is not adapted to the code and saved into the corresponding internal structure until the following code subdivision.

4.2.2.4 Prepare Data

Once the data is loaded, the script's data structure must be filled with the corresponding values for each field. This section of the script's code saves the loaded data in the data structure of the program. The main elements that should be saved in the program's structure after processing the data are:

- XZ plane B-mode 2D matrix.
- YZ plane B-mode 2D matrix.
- XZ plane v_x velocity component 2D matrix.
- YZ plane v_x velocity component 2D matrix.
- XZ plane v_y velocity component 2D matrix.
- YZ plane v_y velocity component 2D matrix.
- XZ plane v_z velocity component 2D matrix.
- YZ plane v_z velocity component 2D matrix.

However, as has been pointed out before, the data loaded in the script needs to be processed before being ready to be saved. Parallel to what occurs in the same code section in the two dimensions visualization, the *Prepare data* section of the three dimensions visualization interpolates both the B-mode and velocity data.

Since this 3D visualization script is actually visualizing two 2D planes in a 3D space and the corresponding 3D velocity values for each plane, no 3D interpolation is needed. This section, thus, performs eight 2D interpolations following the same principles as the homonymous section from the two dimensions visualization script (4.2.1.4).

4.2.2.5 Mask

Following a similar procedure as the one described in section 4.2.1.5 for the 2D script, this section manages the acquisition of the masks that will be used by the program to combine the B-mode and velocity matrices.

Two masks will be produced in this section, one for each cross plane of the data (XZ and YZ). The loading or creation of each of these two mask is done using the same process as followed in the one-plane 2D visualization script.

4.2.2.6 Calculate velocity range and/or vectors maximum

This section of the report has been included with the aim of making the reader notice the non-existence of this section in the 3D visualization script's code. The reason behind not including this section in the 3D script lies in one singularity of the offered visualization: only one frame is visualized.

The homonymous section from the 2D script defined its functionality as the calculation of the velocity range or the maximum value of the velocity vectors for all the frames of a dataset. Since the functionality of this script is to display one data frame (with two planes) under certain visualization parameters, the existence of this section in the 3D script is, thus, meaningless.

However, as the code has been developed in a way where it can be easily adapted to visualizing time-changing datasets with more than one frame, this section can also be implemented if it becomes a demand.

4.2.2.7 Visualization 2D

Once the data is ready for the visualization, a 2D visualization of each plane is performed, with the idea of giving the user a preview of the planes that are going to be visualized in a 3D space. This 2D visualization follows a procedure comparable with the 2D visualization performed in the 2D script.

In this section, the two planes are depicted separately in different figures, having similar visualization possibilities as the offered in the 2D script for a single frame. In this visualization the auxiliary plots have not been implemented, as it has been understood that this script has a different purpose.

4.2.2.8 Visualization 3D

Finally, the 3D visualization itself is performed through this section. The implementation of this code section was subject to the wide range of different visualizations offered by the program also for the 3D module.

As happened with the 2D visualization options, the ones offered in the 3D display can be combined, expanding the visualization possibilities. The following enumerates the 3D display options:

- Visualizing the velocity data with a colormap. As can be observed in figure 4.30.
- Visualizing the velocity data with 3D vectors that emerge from the colormap. As can be observed in figure 4.31.

- Visualizing the researching group logo on the corner of the visualization. As can be observed in figures 4.30 and 4.31.
- Visualizing through the colormap either the absolute velocity or any component of the velocity vectors in the region.
- Visualizing the velocity data using a certain velocity range (introduced by the user) for the velocity colormap.
- Visualizing the velocity data using a certain velocity range (calculated by the program) for the velocity colormap.
- Visualizing the velocity data using a defined value (introduced by the user) as maximum for the vectors in the vectors' visualization.
- Visualizing the velocity data using a defined value (calculated by the program) as maximum for the vectors in the vectors' visualization.
- Visualizing the image choosing the velocity units between: m/s, cm/s and mm/s. As can be observed in the figures presented after the enumeration.
- Visualizing the image choosing the axes' units between: m, cm and mm.
- Visualizing the image choosing the number of labels that appear in the colorbar.
- Visualizing the image choosing the number digits that appear in the colorbar labels.
- Visualizing the velocity vectors determining the size of the matrix that contains its origins.
- Visualizing the data choosing the number of different B-mode values in the gray scale.
- Visualizing the data choosing the number of different velocity values in the velocity color scale.
- Generating a *gif* image that changes the image view by increasing the azimuth of the viewpoint from 0 to 360 degrees around the 3D space, maintaining a constant elevation and radius. Figure 4.32 illustrates the meaning of azimuth and elevation in the coordinate system of the 3D environment's view.
- Generating a video that changes the image view by increasing the azimuth of the viewpoint from 0 to 360 degrees around the 3D space, maintaining a constant elevation and radius. Figure 4.32 illustrates the meaning of azimuth and elevation in the coordinate system of the 3D environment's view.
- Generating the video or *gif* image with a certain azimuth increment between the frames.
- Generating the video or *gif* image with a certain period between the frames.
- Generating the video or *gif* image without visualizing each frame in the process.

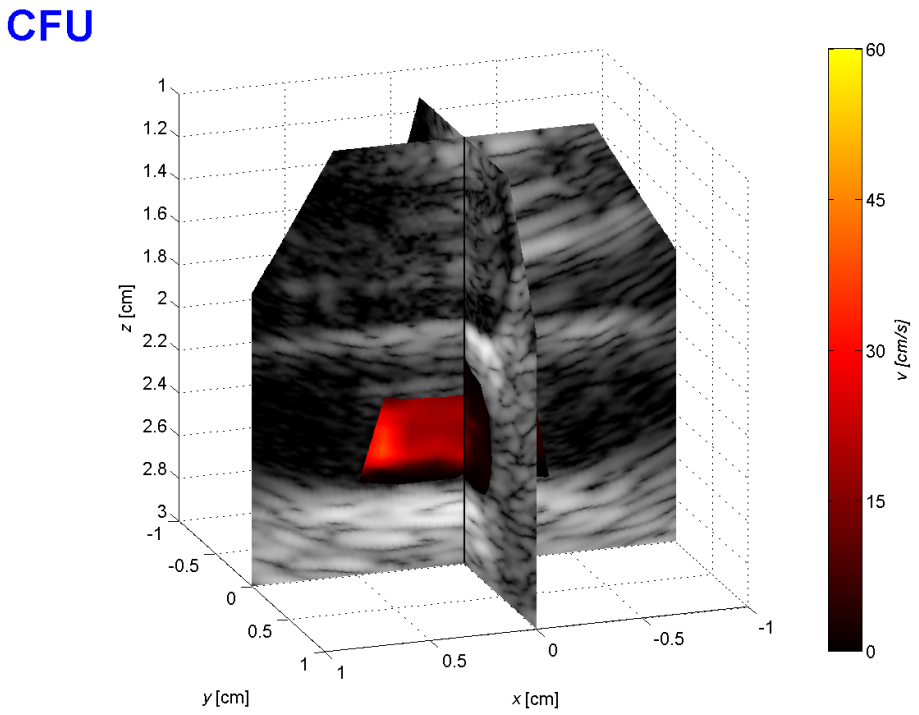


Figure 4.30 – 3D visualization of two cross planes. The colormap visualization shows the value of velocity in each point of the two regions. Data provided by Simon Holbek.

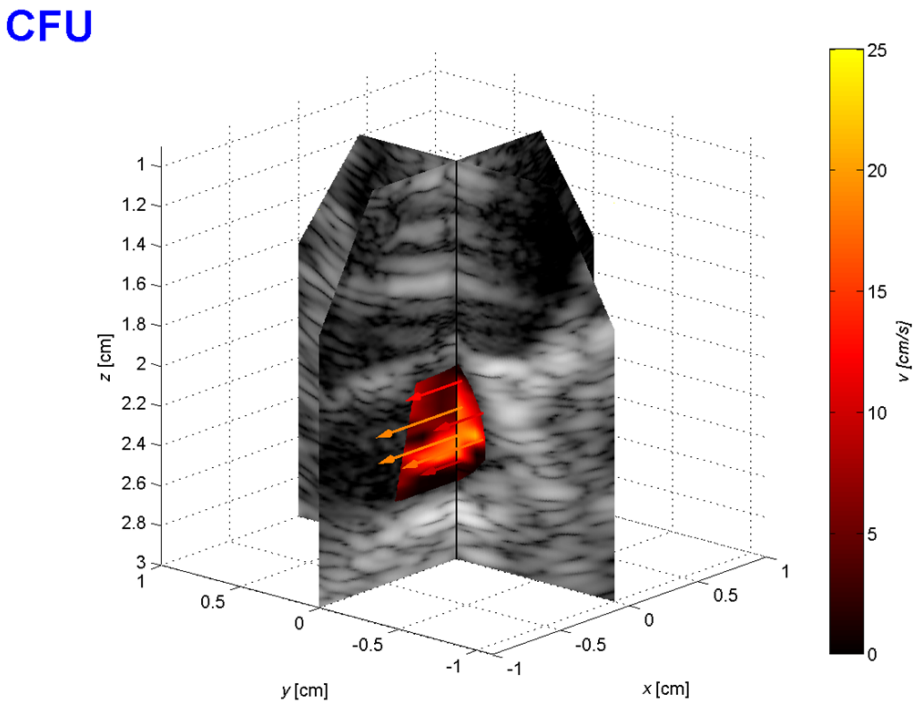


Figure 4.31 – 3D visualization of two cross planes with the vectors representation of the v_x component of the 3D velocity vectors from YZ plane. The colormap visualization shows the value of velocity in each point of the two regions. Data provided by Simon Holbek.

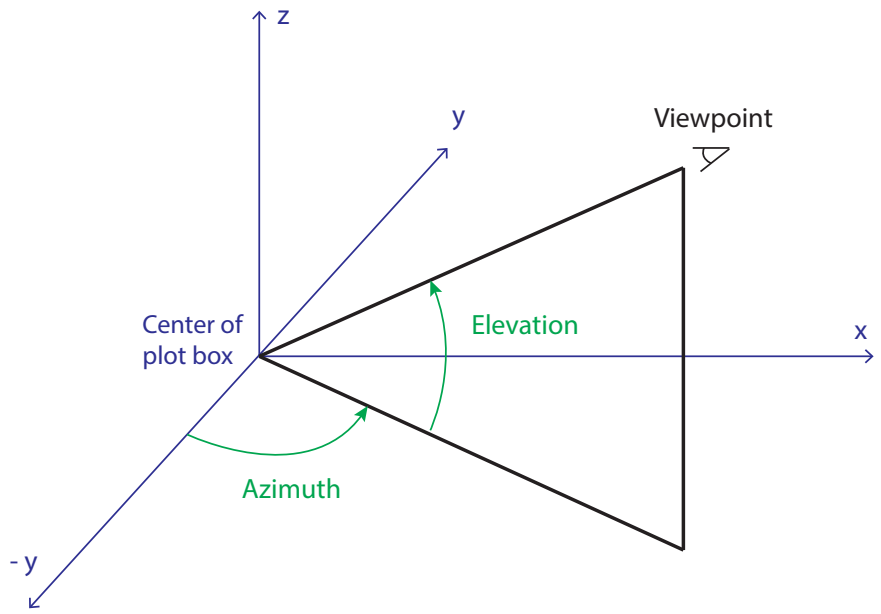


Figure 4.32 – Diagram that illustrates the coordinate system of the 3D environment's *view* in MATLAB. Quoting [46]: *"Azimuth is a polar angle in the x-y plane, with positive angles indicating counterclockwise rotation of the viewpoint. Elevation is the angle above (positive angle) or below (negative angle) the x-y plane"*. Image inspired by the *view* diagram from [46].

The process of yielding the final visualization in the 3D environment can be summarized in the following three steps:

1. Creation of the two cross planes that will be visualized.
2. Visualization of the cross planes in a three dimension space.
3. Generation of a *gif* or a video if it has been requested.

In order to give the reader a deeper understanding of how these steps are accomplished, each of them will be independently described.

Before the creation of the two cross planes, if the user has not selected any visualization range for the colormap, the program calculates the appropriate range for the two planes, in order to use it in the planes formation.

Formation of the two cross planes

The creation of each of the two cross planes that will be visualized in the 3D environment is performed by a function. The code of this function builds the matrix where the static data (B-mode) and the flow data (velocity) are combined.

Due to the nature of this function's task, its code presents similarities with the function that performs the 2D visualization of a frame of data (see pages 41 to 48). For that reason, the following lines will not deepen in the processes already described in the mentioned pages.

Having selected the velocity that will be plotted through the colormap, the code is ready to face the data matrices. The B-mode and velocity matrices are processed in order to be able to build the image matrix of the plane. Once the processing is done and the masks has been applied, the B-mode and velocity matrices are ready to be combined:

1. First an offset is applied to the velocity matrix in order to have all its values above the number of B-mode values.
2. Then both matrices are combined. B-mode values will be in a range from zero to to the number of B-mode values and velocity values in a range from the number of B-mode values to this value plus the number of velocity values.

Figure 4.21 presents a diagram that clarifies the structure of the created matrix. Then, instead of being plotted, this 2D matrix of the plane becomes the output of this function. The colormap and colorbar settings related to the plane are also given back to the main script at the end of this function.

It should be mentioned that neither 2D vectors nor 2D streamlines are created by this function since they are not needed for the 3D visualization of the planes.

Visualization of the cross planes in a 3D space

The visualization of the planes in the 3D space is also performed by a function. The visualization of the two cross planes of data (XZ and YZ) is performed according to the options chosen by the user.

In the same way as the 2D visualization script offers three visualization modes (basic, vectors and streamlines), the 3D visualization allows the user to visualize his data in two main modes:

- A visualization that presents the velocity solely through the colormap.
- A visualization that presents the velocity through the colormap and 3D vectors.

Both visualization modes share large part of the process that leads to their visualization. The following lines will describe this process. First, three arrays that correspond to the visualization axes (x y z) are created under the requirements of the visualization. After doing it, a 3D matrix of the visualization environment is created in order to place on it the received 2D matrices of the cross planes. Once this has been done, the code is ready to visualize the cross planes in the right plane of the 3D environment matrix.

If the user has selected the visualization that includes the 3D vectors, the program has to go through a few additional steps on top of the basic ones. First, the code prepares the three components of the vectors for each plane for the vectors visualization. This step is similar to the analogous one from the 2D vectors visualization, but applied to two planes with three components of the velocity vectors each one. In this case, also, the calculated maximum value of the vectors has been proven to give an non-illustrative visualization. For this reason, the user needs to set a scale factor related to his needs. The visualization setted for the vectors as default visualize only the vectors of YZ plane, using a scale factor of 5, visualizing only the v_x component of the vector. The visualization obtained with this settings has proven to be the most illustrative for the researchers' current needs in the 3D space display. An example of this 3D vectors visualization can be observed in figure 4.31.

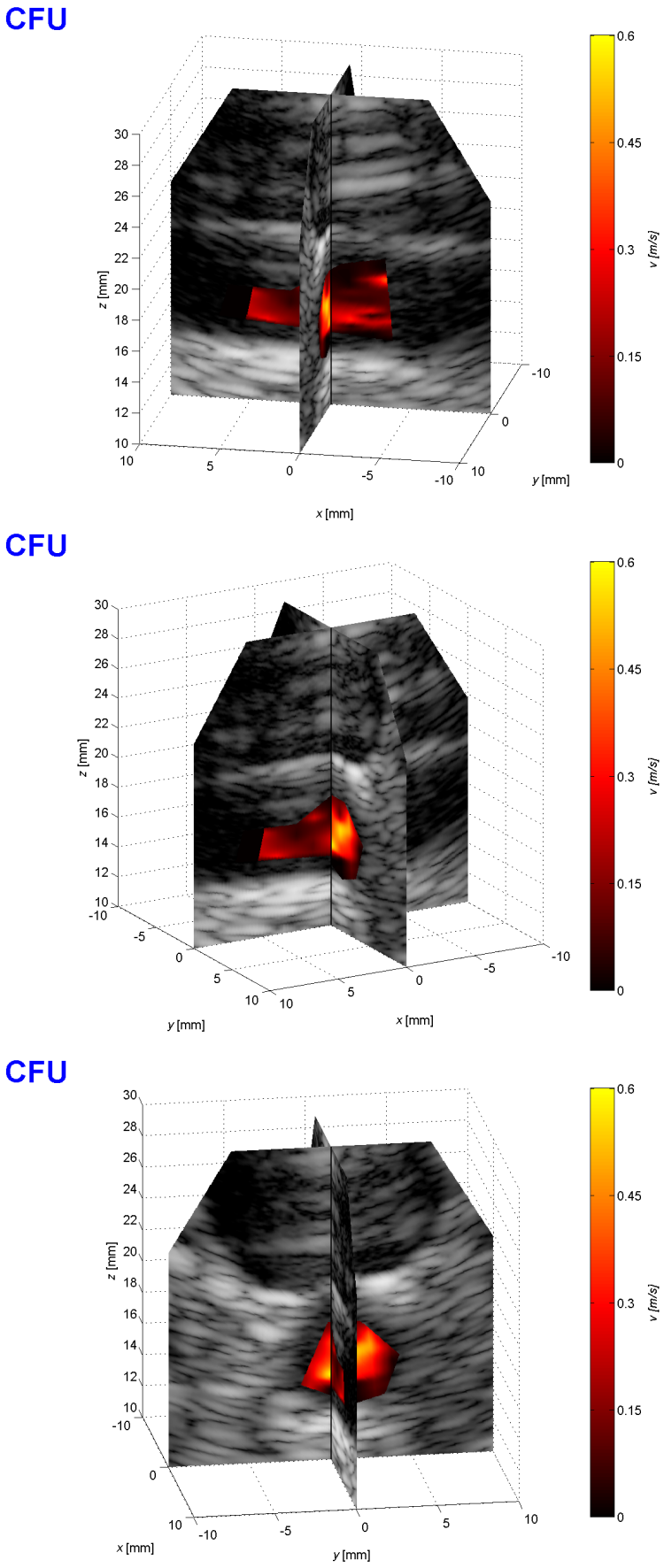


Figure 4.33 – Three different frames of a *gif* that turns the view around the 3D visualization of two cross planes of data. The images correspond, from top to bottom, to the azimuth angles: 190, 150, 80. Data provided by Simon Holbek.

The visualization of these 3D vectors is performed after having the two cross planes visualized in the 3D space. Despite the existence of a MATLAB's function called `quiver3` that visualizes vectors in a 3D space, the function chosen for visualizing vectors with this program is a modification of a function called `mArrow3`. The function `mArrow3` has been developed by Georg Stillfried and can be found in [46]. Quoting his creator, this function *"plots a 3D arrow as patch object (cylinder + cone)"*. The reason behind using this function instead of `quiver3` lays in the researchers preferences for the 3D vectors visualization. In this case, the clear visibility of the arrows has been prioritized over other characteristics of the vectors.

Finally, the colormap and colobar settings received by the previous steps are applied and CFU logo is added to the figure. The function does not provide any output, but generates the final 3D figure.

Gif or video visualization

Lastly, back in the 3D visualization script a *gif* or video is created, if it has been requested.

The process of creating the *gif* or the video is the same as the one followed in the 2D script. However, the frames used to create the *gif* or video of the 3D are different views of the same frame in the 3D display. As has been explained before, the image view is changed by increasing the azimuth of the viewpoint from 0 to 360 degrees around the 3D space. By doing this, the program generates a *gif* or a video that offers a complete view of the 3D visualization. An example of the velocity graphs generated and saved by this function can be observed in figure 4.33.

4.3 Description of the program's structures

This section of the chapter gives an overview of the two structures in which the program functioning is based. These two structures are called `data` and `options`. The following lines will deepen more in `data` due to the relevance of all its fields. Later, `options` will be briefly described, since the different available options of the program have been already presented through the modules' description.

It must be pointed out that neither of the two structures needs to have all its fields filled with information during every execution of the program, as some fields are not needed to the performance of certain data visualizations.

4.3.1 Data

This structure contains the data organized in fields that facilitate its processing and visualization. It is divided in four main fields that are, in turn, divided in fields themselves. The following lines analyse this structure, whose schematic diagram can be observed in figure 4.34.

The program uses two types of data structures depending on the dimensions of the data. There are, thus, a 2D and a 3D versions of the data structure. However, both versions of the structure follow the same organization.

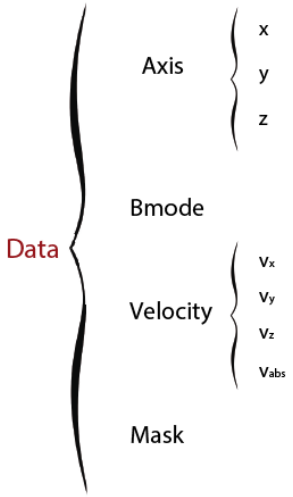


Figure 4.34 – Diagram of the fields of Data structure.

4.3.1.1 2D data

Axis:

The axes of the 2D image are saved in this group of fields. The vertical axis is assigned to the z field, while the horizontal axis can be saved into the x or y field, depending on the reference system selected for the visualized plane. As default, the program assumes the axis data to be in meters, but the user can change it to millimetres or centimetres.

B-mode:

The 2D matrix with the logarithmic compressed values of the B-mode is saved under this field. The logarithmic B-mode values are delimited into the [-60, 0] range.

Velocity:

All the velocity values are saved into 2D matrices under this category. The user can save a matrix for each of the three components of the velocity vectors, as well as one for the absolute value of velocity in each point of the matrix. Some fields can be left unused depending on the available data and the visualization pursued.

Mask:

The program generates the image visualization from the information contained into the B-mode and velocity matrices. To be able to perform this visualization a mask is needed to separate the regions in which the user wants to visualize either B-mode or velocity. This mask is saved as a 2D matrix in this field of the structure, it has ones in the positions which should show velocity and zeros in the B-mode ones.

4.3.1.2 3D data

Axis:

The three axes of the 3D space of visualization are saved into these fields. The two cross planes visualized correspond to XZ and YZ reference planes, so the corresponding horizontal axis (x or y) is used for its generation. As default, the program assumes that the axes' data comes in meters, but the user can change it to millimetres or centimetres.

B-mode:

Two fields that correspond to the two B-mode images, one for each visualized plane. The B-mode logarithmic compressed values are saved in a 2D matrix for each plane, delimited into the [-60, 0] range.

Velocity:

Again, the velocity values are saved into 2D matrices, but in this case there are two velocity matrices of each type, one for the XZ plane and other for the YZ one. The user can save the three components of the velocity vectors as well as the absolute value of velocity for each plane. Some fields can be unused depending on the available data and the visualization pursued.

Mask:

As the 3D visualization corresponds to two 2D planes visualization, two masks (one for each plane) are needed to separate the regions in which the user wants to visualize either B-mode or velocity. These masks are saved as 2D matrices, having ones in the positions which should show velocity and zeros in the B-mode ones.

4.3.2 Options

The program's different options are organized under the structure called `options`. This structure allows the user to control the scripts and change the characteristics of the visualization just by managing its fields. It must be highlighted that, after the visualization, the user can look up all the values of its settings by just checking the fields of this structure. This structure is organized in six different field groups, according to its functionalities, as can be observed in figure 4.35.

4.3.2.1 General

The options under this category configure the main settings of the visualization scripts, such as the data source. But also let the researcher the possibility of choosing to make a video or a *gif* from the chosen data. Also, the operative system of the computer that is running the program is saved in order to automatically adapt the code to it.

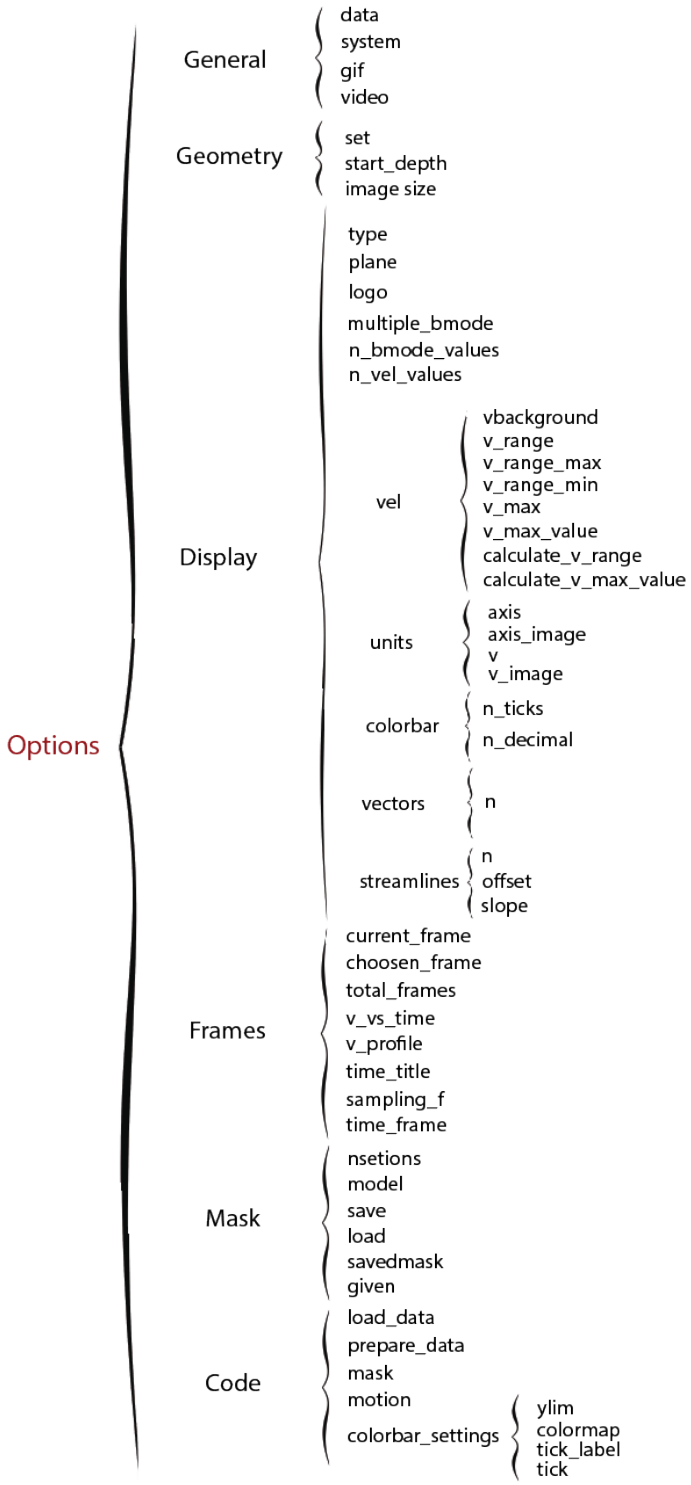


Figure 4.35 – Diagram of the fields of Options structure.

4.3.2.2 Geometry

This group of fields is related to the need of some researchers of changing the interpolation geometry, as has been described in section 4.2.1.4. There are two basic settings that can be easily changed and, thus, have been included in the diagram of figure 4.35. However, other fields are used under this options' category in order to change the remaining interpolation parameters.

4.3.2.3 Display

The options under this category are focused in the pure visualization settings, giving the user the autonomy of choosing every detail of the visualization without having to change the program's code. Some of the options are managed internally by the code in order to provide the visualization requested.

4.3.2.4 Frames

The options related to the visualization of one or more than one frame of a dataset are saved under this category. Also, this group of fields manage the options related to the *gif* and video generation and the auxiliary plots creation.

4.3.2.5 Mask

The group of options under this field are used for the handling of the mask creation and its management once it has been created.

4.3.2.6 Code

The group of options under this category is related to the internal running of the program, providing the user with key information and giving him control over the visualization process.

CHAPTER 5

Evaluation

This chapter describes the program testing and evaluation. This is done through three sections that detail different aspects of the validation performed on this thesis project.

5.1 Testing

With the objective of testing the functioning of the developed program, the program has been tested through the visualization of a wide range of datasets from the researchers of the Center for Fast Ultrasound Imaging in DTU.

These visualizing tests have been performed, first, by the thesis author alone, in order to check the correct functionality of the developed code. Then, testing sessions with the researchers were carried out. The feedback received from this testing sessions was key in the development of the program to its present form. The already discussed diagram presented in figure 4.2 exposes this procedure. As can be understood from the diagram, the testing of the program has been performed through the whole program development. This testing and feedback sessions were specially informative after the development of a new program's feature.

The testing of the program's efficiency was also key in the development of the final version of the code. A large number of modifications have been performed in order to increase the velocity of the script's execution without loosing the characteristics that make it researchers-friendly. It must be kept in mind that the program's velocity and user friendlyness were mayor priorities in the program's design.

After finishing the latest version of the program, an exhaustive checking of the correct functioning of all the program offered features was performed with positive results. It can be concluded, then, that the developed program performs correctly all the visualization possibilities that if offers.

5.2 Output analysis

With the aim of analysing the output of the program, this section goes through different examples of images that are yield by the two scripts of the code.

5.2.1 2D output

Figures 5.1, 5.2, 5.3, 5.4 and 5.5 present the different 2D visualizations with indications for a better understanding of the relations between the code and the output.

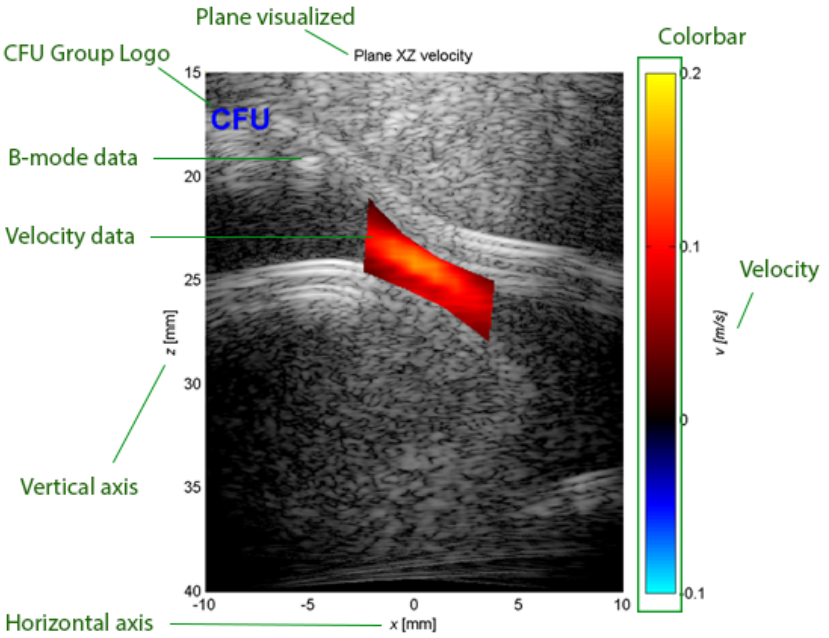


Figure 5.1 – Elements of the basic visualization of the velocity data (colormap). Data provided by Carlos Armando Villagómez Hoyos.

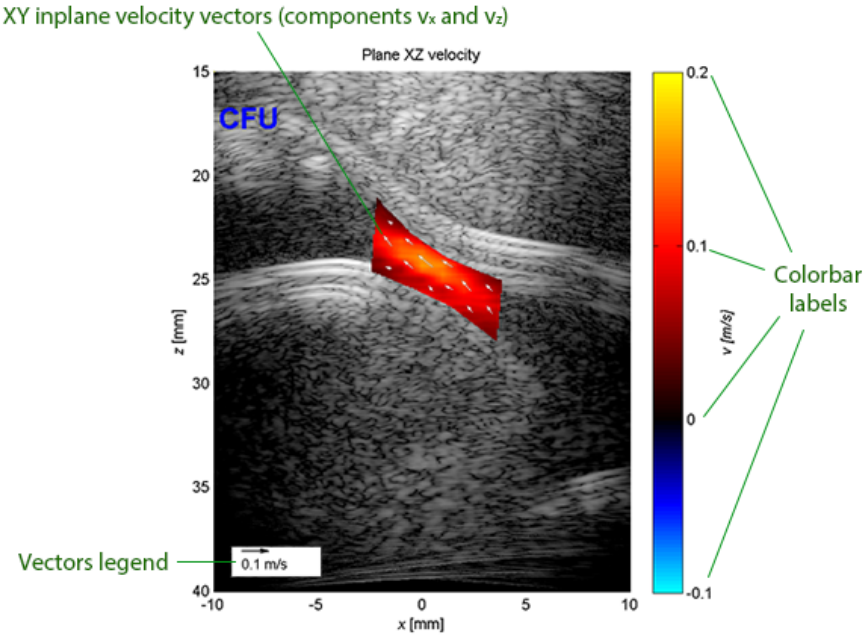


Figure 5.2 – Elements of the velocity data visualized through vectors and colormap. Data provided by Carlos Armando Villagómez Hoyos.

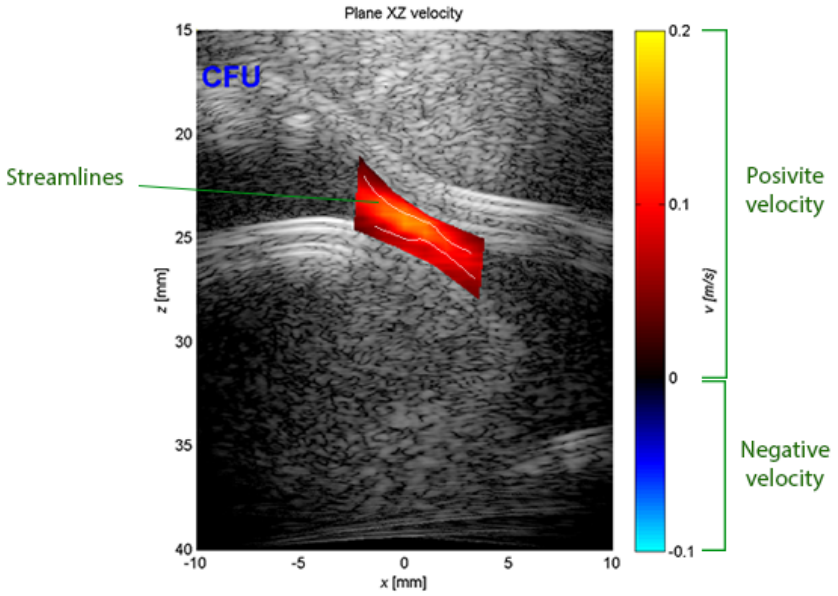


Figure 5.3 – Elements of the velocity data visualized through streamlines and colormap. Data provided by Carlos Armando Villagómez Hoyos.

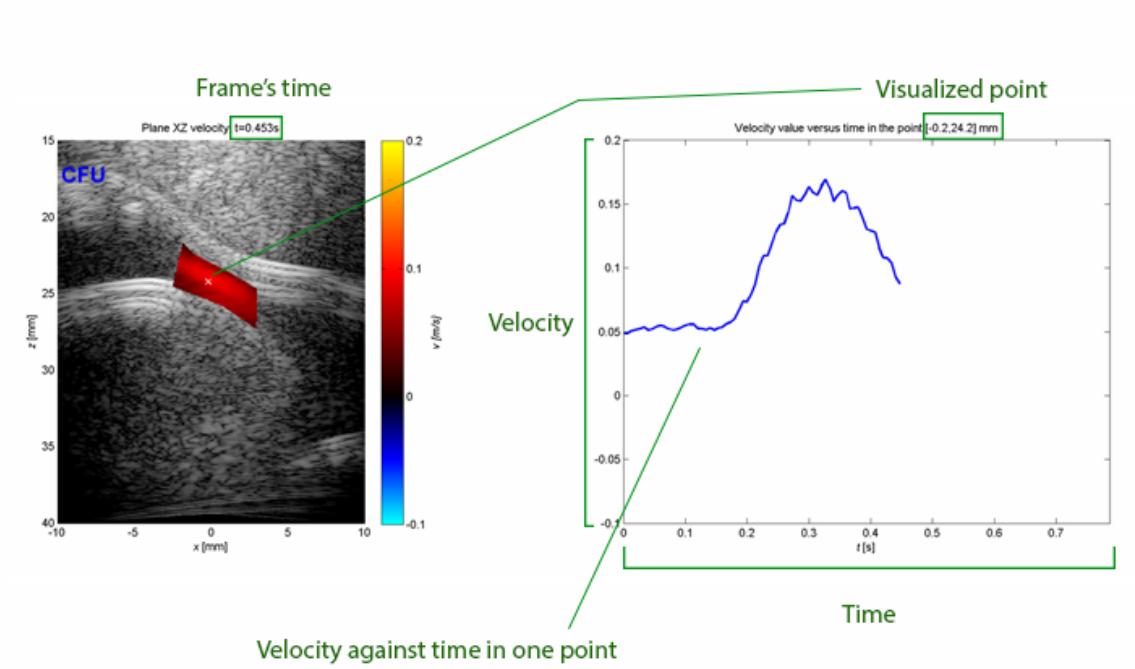


Figure 5.4 – Elements of the visualization of the velocity in one point against time, using an auxiliary plot. Data provided by Carlos Armando Villagómez Hoyos.

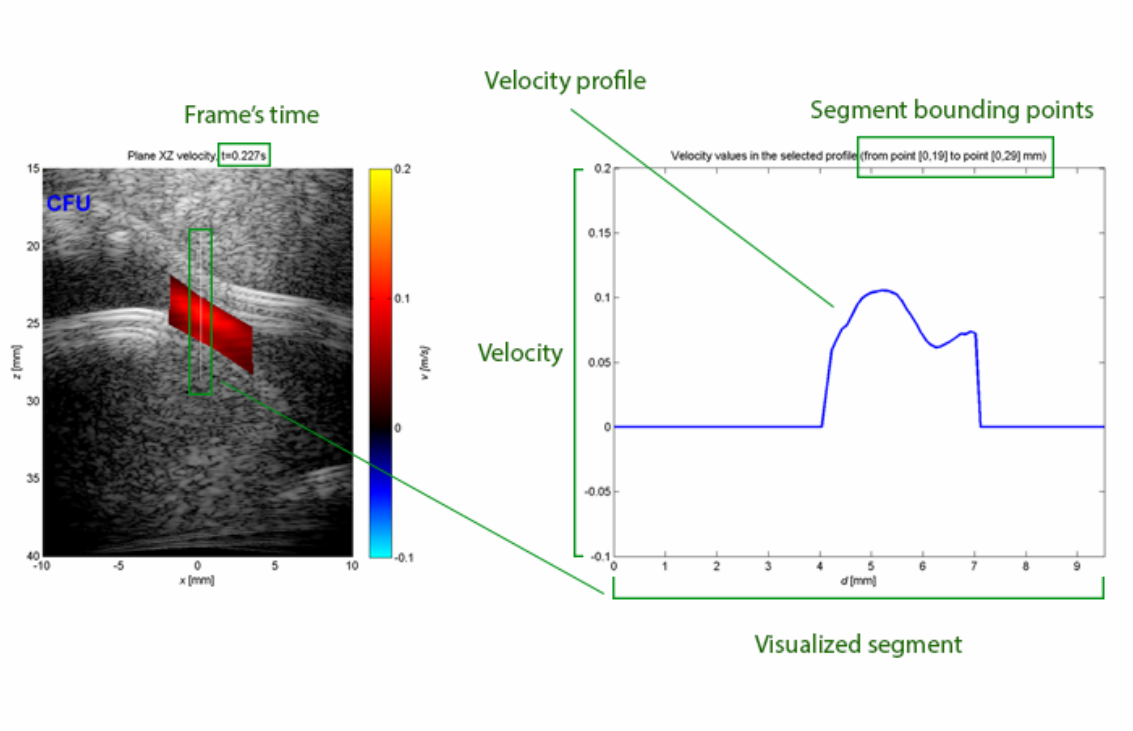


Figure 5.5 – Elements of the visualization of the velocity values in one segment of the image, using an auxiliary plot. Data provided by Carlos Armando Villagómez Hoyos.

5.2.2 3D output

Figures 5.6 and 5.7 present the different 3D visualizations with indications for a better understanding of the relations between the code and the 3D output.

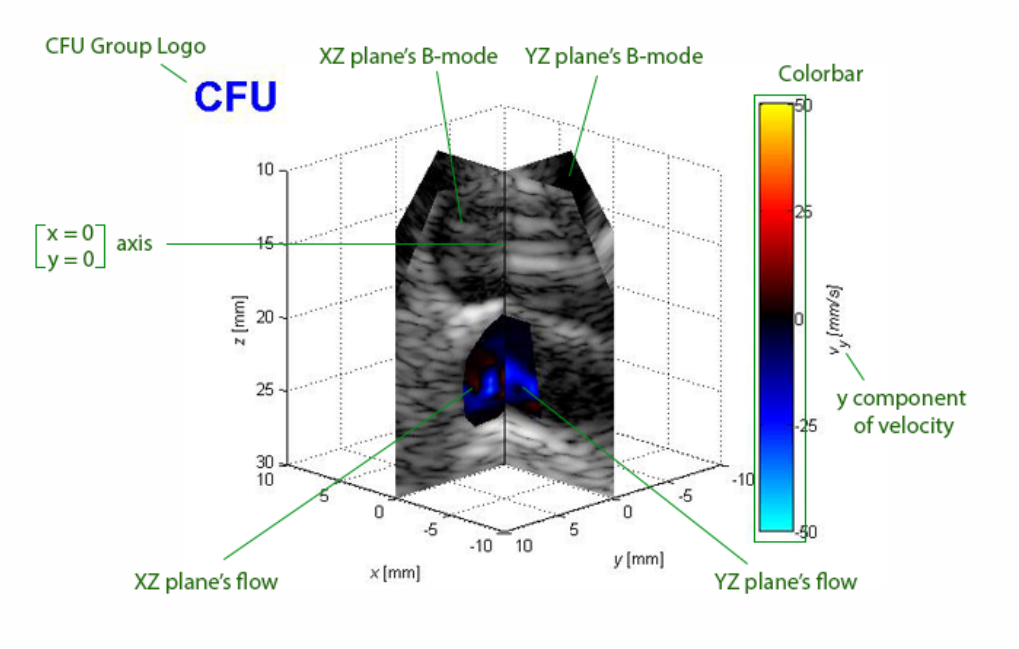


Figure 5.6 – Elements of the 3D visualization of two cross planes. Data provided by Simon Holbek.

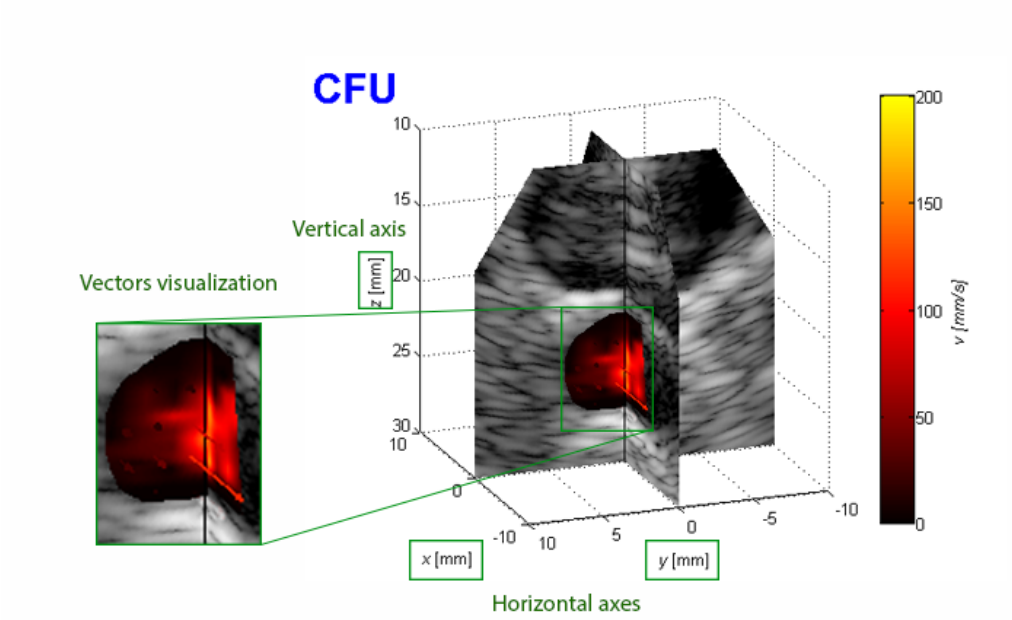


Figure 5.7 – Elements of the 3D visualization of two cross planes with the vectors representation of the v_x component of the 3D velocity vectors from YZ plane. Data provided by Simon Holbek.

5.3 Processing time analysis

This section of the chapter is conceived to study the quickness of the visualization scripts of the program. The section has been divided in two subsections that correspond to the two different visualization scripts: 2D visualization and 3D visualization.

5.3.1 2D visualization

With the aim of having an idea of the percentage of total execution time that is used in each code section, the 2D visualization code was executed several times with datasets from different sources. From each execution of a dataset, the total duration of a visualization was saved, saving also the duration of each of the code sections that lead to it. Then, the percentage of the total time employed by each section was calculated. The values presented in table 5.1 are the result of making an average of these calculated percentages.

From the results presented in the table it can be highlighted that the first two subdivisions of the script do not have an effect in the duration of the visualization process, however, the *Visualization 2D* section takes more than half of the time of the total visualization. Beyond the fact that this is the section that performs the visualization itself, this high percentage of time is due to the effect of the visualizations that involve auxiliary plots, *gif* or video generation. The *Mask* section also takes a high percentage of the time, but in this case this circumstance is due to the effect of the manual selections of the mask that depend exclusively on how fast is the user at drawing a proper region in the image. The executions of the program in which the mask is loaded as data, the percentage of time use by this code section is reduced to a minimum. It must be taken into account that *Load Data* and *Prepare Data* take considerable processing time, even though they are smaller than the others mentioned here. The high value of the *Prepare Data* section is due to the interpolation of velocity data needed in some of the data sources. The time used for *Calculate velocity range and/or vectors*

max is not high, but it is better to avoid using this section every time, in order to gain some processing speed.

Table 5.1 – 2D visualization script processing time relation between its sections.

Script section	**Time [%]**
Input	0,0056
Default	0,0109
Load Data	6,0738
Prepare Data	12,9013
Mask	18,7129
Calculate velocity range and/or vectors max	3,8019
Visualization 2D	58,4936

However, it should be highlighted that among the options of the program there are three which allow the user to avoid the execution of any of the following sections:*Load Data*, *Prepare Data* or *Mask*. This can be done only if the code has been previously executed with the same data. By doing this the users can reduce the total execution time to around 60 % of the total. Also, the section *Calculate velocity range and/or vectors max* can be easily skipped by selecting a velocity range for the visualization.

5.3.2 3D visualization

Following the same procedure as with the 2D visualization scrip, the 3D visualization code was executed several times with datasets from different sources. Then, parallel calculations as the described for the 2D script were done, with the aim of having an idea of the percentage of total execution time that is used in each code section. Table 5.2 presents the corresponding results.

Table 5.2 – 3D visualization script processing time relation between its sections.

Script section	**Time [%]**
Input	0,0001
Default	0,0224
Load Data	3,5859
Prepare Data	3,4216
Mask	24,3866
Visualization 2D	2,2634
Visualization 3D	66,3199

Looking at the table, it is notable how the *Visualization 3D* section takes almost 70 % of the total time. The reason behind it is the amount of time that the *gif* creation takes, turning the view around the 3D space. It must be highlighted that the 2D visualization of the two planes takes a small percentage of the total time. The *Mask* section takes a notable percentage of the time due to the effect due to the human participation on the process. The executions of the program in which the mask is loaded as data, the percentage of time use by this code section is reduced to a minimum. Again, the loading and processing of the data consume a

relatively small part of the time due to the fact that only one frame is loaded and processed in this script. In this case, *Input* and *Default* also take a negligible percentage of the time.

As happened with the 2D visualization script, it should be highlighted that three program options allow the user to avoid the execution of any of these sections: *Load Data*, *Prepare Data* or *Mask*.

After the the study of the data used in the development of this project, it must be pointed out that the processing time of the data handled by the program depends largely on the size of the visualized data. However, the time used in the program's execution can be shortened by using the program's options. All in all, the efficiency in the visualization is one of the main objectives pursued in this project.

CHAPTER 6

Conclusions and Work Ahead

The proven importance of ultrasound systems in the study of blood flow as well as for the clinical diagnosis of diseases of the circulatory system, infers that advances in this field are significant for the medical community.

The aim of this project consists on helping the researchers in this area by facilitating the visualization process of their research advances using ultrasound systems. Thus, the development of such a visualization tool would afford a more productive use of time for these researchers, which would in turn lead to an enhanced rate of useful advances in this field.

The conclusion of this thesis should be ascertained from the capability of the developed program to fulfil the researchers' needs with regards to the visualization of their investigations. Under this premise, the following discussion analyses the objectives achieved in the project development as well as the outcomes reached after its implementation.

As have been analysed in section 5.3 of the thesis, the duration of the visualization process for certain data depends largely on the size of the data. More specifically, it depends on the size of the matrices processed and, above all, on the number of velocity frames of the dataset. Due to this, each step of the visualization takes a longer time when facing larger datasets. However, the step-based structure of the visualization scripts allows the possibility to shorten, and thereby optimize, the process duration in successive visualizations of the same dataset.

Another objective pursued in the development of such a program was the possibility of an easy visualization that entails the use of this group of scripts as a logical decision when a visualization of data is needed. In order to achieve this idea, the program was developed with a number of default settings that cover all the visualization aspects. Moreover, a visualization can simply be yielded by providing the data and the name of the data source. Furthermore, even though this script does not include a graphic interface, no prior programming knowledge is required to use it. On top of that, having all the main information of the visualization saved in two structures, `data` and `options`, allows the user to access it without needing to look deep into the code.

Knowing that the prospective users of this program are researchers that work with MATLAB, the development of the code in this language as well the documentation created about program must allow these researchers to add new functionalities to the code if it is necessitated in future investigations. The mentioned documentation is composed by: the User Manual and the Program's Detailed Description.

All in all, the implementation of this code was performed with the goal of trying to find an equilibrium between a fast visualization and an accessible process to achieve this visualization, which would be understandable by users with a background in MATLAB. As a consequence of this equilibrium pursuit, the developed program only finds its limitations with increased size of datasets. The main issue related to the size of data comes from the incapability of the RAM from a researcher's regular computer to manage data with the size of several gigabytes.

In the search for this equilibrium in the creation of a useful program, feedback from these researchers has been decisive. The program basic visualization capacity, together with auxiliary visualization modes and the features suggested by the researchers, has promoted the use of visualizations obtained with this tool to the regular use in the Center for Fast Ultrasound Imaging. In this way, the researchers use it to facilitate their diary works and share results with their peers. This can be considered an important indication to the success of the project, which would be realized if the program beeps being regularly used by researchers in their investigations in this field.

6.1 Future work

With the objective of continuing the pursuit of this research path that involves a visualization that covers the present needs of researchers, the potential implementation of new features in the program must be considered. The following suggests future work to improve the results achieved with this thesis project.

As has been already said, the design of the program was developed for an easy implementation of any new functionality, allowing a user to add functions to the program if new visualization needs arise.

Beyond the development of possible new visualization functionalities, especially in 3D visualization (an emerging field that has a long way to go in overcoming many challenges), the program finds room for improvement when facing largr datasets. In this way, datasets of a several gigabytes cannot be managed by a regular computer's RAM, thus hindering the visualization.

Once this situation was detected, this thesis author developed a code that overcame the issue. The code consists of a loop that loads, processes and saves each frame (where each frame has a reasonable size) from a dataset of several gigabytes. Once all the frames are ready, a video and a *gif* image can be created. The total time spent over the whole process is close to the time that would have spent the program developed in this thesis if the computer was capable of handle the large dataset. However, the time required for subsequent visualizations using this code will always be the same, since the time of processing will not be economized by building on previous visualizations. As can be deducted, this fails in the capability of offering a fast succession of different visualizations of the same dataset. Thus, this functionality has not been implemented in this thesis' program, since it would disallow its flexible and fast visualization capabilities, which are important to match the project objectives. However, the idea behind the code for visualizing large datasets should be considered for future improvements in the software that may achieve to overcome this inconvenient situation when handling large datasets.

Furthermore, it ought to be considered that one of the principal design decisions taken in this project was to

neglect a graphic interface. As such, the program code is exposed to the user during execution, allowing the user to read it if required. However, if researchers require this functionally, the development of a graphic interface for the scripts would make the visualization process even more simple.

With all this in mind as future prospects to build upon the current project, this thesis must be seen not only as a functioning program that is allowing researchers to visualize the data of their investigations, in the present, but also a strong base of code which may be adaptable to future visualizations.

References

[1] E.P. Widmaier, H. Raff and K.T. Strang. *Vander's Human Physiology*. McGraw-Hill. 2011.

[2] J.A. Jensen. *Estimation of Blood Velocities Using Ultrasound: A signal Processing Approach*. Technical University of Denmark, Department of Electrical Engineering. Denmark. 2008.

[3] J.L. Prince and J.M. Links. *Medical Imaging. Signals and Systems*. Pearson Prentice Hall Bioengineering. 2006.

[4] K.L. Hansen, J. Udesen, F. Gran, J.A. Jensen, and M. B. Nielsen. Invivo examples of complex flow patterns with a fast vector velocity method. *Ultraschall in Med*, 30:471–476. 2009.

[5] W.F. Boron and Emile L. Boulpaep. *Medical Physiology*. Elsevier Health Sciences. 2008.

[6] L. Sherwood. *Human Physiology: From Cells to Systems*. Brooks/Cole Cengage Learning. 2012.

[7] W. Nichols, M. O'Rourke and C. Vlachopoulos. *McDonald's Blood Flow in Arteries*. CRC Press. 2011.

[8] L. Boxt, S. Abbara, and S.W. Miller. *Cardiac Imaging: The Requisites*. Mosby. 2009.

[9] P. Heintzen, W. E. Adam. History of cardiovascular imaging procedures. *Zeitschrift Fur Kardiologie*, 91(4):64-73. 2002.

[10] J. Nielsen. Iterative User-Interface Design. *IEEE Computer*, 26(11): 32-41. 1993.

[11] J.L. Nath. *Using medical terminology: a practical approach*. Lippincott Williams & Wilkins. 2006.

[12] V.B. Ho and G.P. Reddy. *Cardiovascular Imaging*. Elsevier. 2011.

[13] R.K. Harris and Roderick E. Wasylishen. *Encyclopedia of NMR*. Wiley. 2012.

[14] H.L. Blumgart, C.Y. Otto. Studies on the velocity of blood flow. *J Clin Invest*, 4: 1–13 1926.

[15] S. Carlsson A glance at history of nuclear medicine. *Acta Oncologica*, 34(8):1095-1102. 1995.

[16] D.J. Higham and Nicholas J. Higham *MATLAB Guide*. SIAM. 2005.

[17] Oge Marques *Practical Image and Video Processing Using MATLAB*. Wiley-IEEE Press. 2011.

[18] J.M. Reid. Doppler Ultrasound. *IEEE Engineering in Medicine and Biology Magazine*, 6(4):14-17. 1987.

[19] J.R. Roelandt. Seeing the heart; the success story of cardiac imaging. *European heart journal*, 21(16): 1281-8. 2000.

[20] L. Durcan. Lumière or An Accounting of the Events of December 28th, 1895 at the Grand Café, Boulevard des Capucines, Paris *Antigonish Review*, 135:21. 2003.

[21] R.A. Meyer. History of ultrasound in cardiology. *J Ultrasound Med*, 23(1):1-11. 2004.

[22] Doppler effect. *The Columbia Encyclopedia, 6th Edition*. 2013.

[23] K. Kirk Shung. *Diagnostic Ultrasound: Imaging and Blood Flow Measurements*. CRC Press. 2005.

[24] T. Kimpe and T. Tuytschaever. Increasing the Number of Gray Shades in Medical Display Systems — How Much is Enough? *Journal of Digit Imaging*, 20(4):422–432. 2007.

[25] D.H. Evans, J.A. Jensen, and M.B. Nielsen. Ultrasonic colour Doppler imaging. *Interface Focus*, 1(4):490–502. 2011.

[26] M.J. Pihl, *3D vector flow imaging*. PhD thesis. Center for Fast Ultrasound Imaging, Technical University of Denmark, 2800 Lyngby, Denmark. 2012.

[27] J. Udesen and J.A. Jensen. Investigation of transverse oscillation method *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 53(5):959-971. 2006.

[28] J.A. Jensen, H. Holten-Lund, R.T.Nilsson, M. Hansen, U.D. Larsen, R.P. Domsten, B.G. Tomov, M.B. Stuart, S.I. Nikolov, M.J. Pihl, Y. Du, J. Rasmussen, M.F. Rasmussen. SARUS: A Synthetic Aperture Real-Time Ultrasound System. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 60(9):1838-1852. 2013.

[29] Velocity. *The Columbia Encyclopedia, 6th Edition*. 2013.

[30] M. Hazewinkel. Trigonometric functions. *Encyclopedia of Mathematics*. Springer. 2001.

[31] D. H. Evans, W. N. McDicken. *Doppler Ultrasound, Physics, Instrumentation, and signal processing*. John Wiley & Sons. 2000.

[32] Illustrative. *Oxford Dictionary of English (3 ed.)*. Oxford University Press. 2010.

[33] Computational program. *The Columbia Encyclopedia, 6th Edition*. 2013.

[34] DTU in Profile 14. *Technical University of Denmark*. 2014.

[35] S.I. Nikolov, H. Holten-Lund, R.T. Nielson, M. Hansen,U.D. Larsen, J.A. Jensen, B.G. Tomov, M.B. Stuart. Performance of SARUS: A synthetic aperture real-time ultrasound system. *IEEE Ultrasonics Symposium*, 305-309. 2010.

[36] J. J. Flaherty, K. R. Erikson, V. M. Lund. Synthetic aperture ultrasound imaging systems. *United States Patent 3,548,642, 1967*. Published 22 Dec 1970.

[37] J.A. Jensen, M.B. Stuart. An Architecture and Implementation of Real-time Synthetic Aperture Compounding with SARUS. *IEEE Ultrasonics Symposium*, 1044-1047. 2011.

[38] J.A. Jensen, M. Hansen, B.G. Tomov, S.I. Nikolov, H. Holten-Lund. System Architecture of an Experimental Synthetic Aperture Real-time Ultrasound System. *IEEE Ultrasonics Symposium*, 636-640. 2007.

[39] C. Villagomez Hoyos, M.B. Stuart, J.A. Jensen. Increasing the Dynamic Range of Synthetic Aperture Vector Flow Imaging. *Proceedings of Spie Medical Imaging*, 9040. 2014.

[40] S. I. Nikolov. *Synthetic aperture tissue and flow ultrasound imaging*. PhD thesis. Center for Fast Ultrasound Imaging, Technical University of Denmark, 2800 Lyngby, Denmark. 2001.

[41] S. Holbek, M.J. Pihl, C. Ewertsen, M.B. Nielsen, J.A. Jensen. 3-D Velocity Estimation for Two Planes in vivo. *IEEE Ultrasonics Symposium*. 1706-1709. 2014.

[42] S.I. Nikolov, M.C. Hemmsen, J.A. Jensen, M.M. Petersen, M.B. Nielsen. Ultrasound Image Quality Assessment: A framework for evaluation of clinical image quality. *Progress in Biomedical Optics and Imaging*, 7629(1):76290C. 2010.

[43] R. Fisher, S. Perkins, A. Walker, E. Wolfart. *HIPR Hypermedia Image Processing Reference*. John Wiley & Sons. 2003.

[44] Streamline. *Encyclopædia Britannica*. 2015.

[45] Medical Solutions. *Siemens Healthcare Global*. Available at: http://www.healthcare.siemens.com/ [online] [Accessed 2014].

[46] Matlab for Technical Computing. *MathWorks.* Available at: http://www.mathworks.com/ [online] [Accessed 2014].

Appendices

APPENDIX **A**

Introducción

De media, el cuerpo humano contiene en torno a 5,5 litros de sangre [1]. Este volumen sanguíneo es transportado a través del cuerpo por el sistema circulatorio, que tiene la responsabilidad de proporcionar oxígeno y nutrientes a los órganos y eliminar el dióxido de carbono y los desechos metabólicos que las células expulsan.

La importancia del sistema circulatorio en la salud del ser humano ha propiciado la creación de una especialidad médica para el estudio específico del sistema vascular, la angiología. Por consiguiente, será necesario dotar a la medicina vascular de las herramientas apropiadas para el diagnóstico de enfermedades relativas al sistema circulatorio.

En este contexto, la relevancia de la ultrasonografía en el estudio de enfermedades que afectan a la circulación sanguínea debe ser resaltada. La ultrasonografía médica se considera un sistema de imagen médica sin riesgo, lo que combinado con su no invasividad, la convierte en apropiada para el estudio de flujo sanguíneo. Esto es posible gracias al uso del efecto Doppler, como se indica en [18]. La información sobre el comportamiento del flujo sanguíneo permite a los médicos la detección de ciertas enfermedades y, por consiguiente, actuar en consecuencia. Por otro lado, se ha de tener en cuenta que la velocidad del flujo sanguíneo tiene dependencia temporal y espacial [4], por ello, la estimación simultánea de sus tres componentes espaciales supone un avance para el desarrollo de las investigaciones actuales en este campo.

El trabajo de desarrollo de nuevos métodos de estimación de velocidad y la mejora de los ya existentes tienen el objetivo de producir representaciones más precisas del flujo sanguíneo y sus complejos patrones. Durante el desarrollo de estos métodos, una rápida visualización de los datos estimados por los investigadores facilitaría la evaluación de los avances realizados. La mencionada visualización debe ser vista como parte del desarrollo de los algoritmos de estimación, y previa al proceso de creación de imagen clínica que se realiza en los hospitales.

Dicho esto, el objetivo del presente proyecto es el diseño e implementación de una herramienta que provea con rapidez una visualización ilustrativa de la velocidad del flujo sanguíneo. En la figura A.1 se muestra un diagrama que ilustra este procedimiento. Se ha de tener en cuenta que los valores de velocidad del flujo sanguíneo son obtenidos con distintos estimadores de velocidad, que son ejecutados en diferentes escáneres médicos de ultrasonidos con equipamiento y parámetros variables. Es por ello que los datos sin procesar presentan una gran variedad de formatos y no hay un método homogéneo de visualización de los mismos. Por consiguiente, la existencia de una herramienta que solvente esta situación se convierte en una necesidad.

Debido a esto, el trabajo realizado en el presente proyecto ha de ser visto como un proyecto de ingeniería que solventa una necesidad práctica existente.

Figure A.1 – Diagrama del rol de el presente proyecto en la proceso de trabajo de los investigadores.

Esta memoria presenta, en primer lugar, los conocimientos contextuales necesarios para la comprensión del proyecto, cuya descripción se desarrolla en profundidad en los capítulos restantes. Con el objetivo de dar al lector una visión general del trabajo realizado, este capítulo aborda la motivación y objetivos que dan sentido al proyecto.

A.1 Motivación

El objetivo principal de este proyecto está motivado por las necesidades de visualización experimentadas por los investigadores que trabajan en la mejora de las técnicas de cálculo de la velocidad del flujo sanguíneo. Como se detalla en el capítulo 2 de esta memoria, el sistema cardiovascular es una intricada red de vasos que se ocupan del transporte de la sangre a través del cuerpo humano. Debido a la complejidad de la dinámica del flujo sanguíneo a través del cuerpo y su interacción con los campos de ultrasonido, se han propuesto diferentes técnicas para superar los retos que este campo de estudio presenta [26]. Los investigadores que trabajan en el desarrollo de estas técnicas necesitan visualizar los resultados obtenidos en sus investigaciones, para evaluar los resultados y compartir sus avances.

El objetivo de este proyecto es, por lo tanto, desarrollar una herramienta que proporcione un método apropiado para la visualización de los vectores de velocidad obtenidos por los investigadores.

Con el objetivo de ser más específico, se ha de definir el carácter de la herramienta que resuelva esta necesidad. De acuerdo con la definición que aparece en La Enciclopedia Columbia [33], a una secuencia ordenada de instrucciones computacionales necesaria para alcanzar una solución se le llama programa computacional. Esto implica que la herramienta para resolver la problemática propuesta, computacionalmente, puede denominarse programa a partir de este punto de la memoria del proyecto.

Con el fin de facilitar el trabajo de los investigadores, el desarrollo de dicho programa deberá llevarse a cabo en el entorno de procesado de datos que ellos utilizan. Habiéndose identificado MATLAB como un programa de cálculo numérico ampliamente utilizado por los investigadores, tanto en las universidades como en la industria [16], el diseño de una herramienta de visualización basada en el lenguaje de programación de este software se presenta como una consecuencia lógica.

Como se indica en [17]: *"Matlab es una herramienta de análisis de datos, creación de prototipos y visualización, con soporte incorporado para matrices y operaciones con matrices, excelentes capacidades*

gráficas, y un lenguaje de programación y un entorno de desarrollo de alto nivel". Sin embargo, el gran potencial de esta herramienta podría verse mermado por la complejidad de uso de algunas de sus funciones. Debido a ello, la herramienta de visualización deseada deberá ser capaz de ofrecer a los investigadores una manera rápida y fácil de generar representaciones visuales de sus resultados, permitiéndoles centrarse en las tareas propias de su investigación.

Se puede concluir, por tanto, que el objetivo de este proyecto es el diseño y desarrollo de un programa escrito en MATLAB que sea capaz de visualizar los datos obtenidos por los investigadores que trabajan en la estimación de la velocidad del flujo sanguíneo. El desarrollo de un programa de estas características implicará el desarrollo de funcionalidades relacionadas con campos como la interpolación de datos, el procesado de vectores, el tratamiento de imágenes y la generación de video.

A.2 Estructura

El presente proyecto se divide en seis capítulos. La siguiente enumeración aporta una breve descripción sobre cada capítulo a partir de la introducción:

- **Capítulo 2** contextualiza el trabajo realizado, describiendo el sistema cardiovascular y los sistemas de imagen médica que se han usado para su visualización. Además, este capítulo también analiza el estado del arte de la visualización del flujo sanguíneo con ultrasonidos.
- **Capítulo 3** describe el proceso que ha conducido al diseño actual del programa desarrollado, analizando las especificaciones del programa y su diseño e implementación. Además, este capítulo también analiza la procedencia de los datos que recibe el programa.
- **Capítulo 4** da al lector una visión general del proceso de implementación del programa, mostrando las funcionalidades del mismo a través de la descripción de sus partes.
- **Capítulo 5** explica cómo se ha realizado la evaluación del programa, abordando además el análisis de los resultados obtenidos en las pruebas de ejecución del los scripts.
- **Capítulo 6** sintetiza las conclusiones alcanzadas a través de la realización del proyecto y presenta posibles mejoras para trabajo futuro.

APPENDIX **B**

Conclusiones y trabajo futuro

El reconocimiento dado a la importancia de la ultrasonografía en el estudio del flujo de sangre, así como para el diagnóstico clínico de las enfermedades del sistema circulatorio, infiere que los avances en este campo tienen una repercusión relevante para la comunidad médica.

El objetivo de este proyecto era ayudar a los investigadores en este área, facilitando el proceso de visualización de sus avances. Por tanto, el desarrollo de una herramienta que realice este tipo de visualización permitiría que estos investigadores hicieran un uso más productivo de su tiempo, lo que a su vez conduce a una tasa mayor de avances en su campo.

La conclusión de este proyecto vendrá determinada por la capacidad del programa desarrollado para satisfacer las necesidades de los investigadores con respecto a la visualización de sus datos. Bajo esta premisa, los siguientes párrafos presentan una discusión que analiza las metas conseguidas con la realización de este proyecto, así como los resultados obtenidos tras su finalización.

Como se indica en el análisis realizado en la sección 5.3 de esta memoria, la duración del proceso de visualización de los datos depende en gran medida del volumen de los datos. Más específicamente, depende del tamaño de las matrices procesadas y, sobre todo, del número de muestras (*frames*) que contenga el set de datos si las mediciones incluyen el parámetro tiempo. Debido a esto, el tiempo de ejecución de cada etapa del programa de visualización aumenta conforme crece el volumen de datos a procesar. Sin embargo, la estructura basada en etapas con la que se han diseñado los *scripts* de visualización permite la posibilidad de acortar, y de ese modo optimizar, la duración del proceso en visualizaciones sucesivas del mismo conjunto de datos.

Otro objetivo perseguido en el desarrollo del programa era facilitar la visualización de manera que su utilización fuera la decisión más conveniente llegada la hora de visualizar este tipo de datos. Con el fin de lograr esa meta, el programa se ha equipado con una configuración por defecto que contempla todos los aspectos de la visualización. De esta forma, el proceso de visualización puede reducirse simplemente a la creación de una imagen ilustrativa siendo necesario proporcionar únicamente un set de datos y el nombre de la fuente de datos. Por otra parte, a pesar de que los *scripts* desarrollados no cuentan con una interfaz gráfica, no se requiere ningún conocimiento de programación previo para su uso. Además, el hecho de tener toda la información relevante de la visualización guardada en dos estructuras, `datos` y `opciones`, permite que el usuario pueda acceder a ella sin necesidad de tener que acceder al código.

Sabiendo que los usuarios potenciales del programa serán investigadores que trabajan con MATLAB, el que el código del programa se haya desarrollado en este lenguaje y se haya realizado una documentación para el mismo posibilitará que estos investigadores puedan añadir nuevas funcionalidades al código en el caso de ser necesario en futuras investigaciones. La documentación mencionada se compone de: un Manual de Uso y una Descripción Detallada del Programa.

Con todo, la implementación de código se realizó con el propósito de encontrar un equilibrio entre una visualización rápida y un proceso accesible para alcanzar la misma, que fuera comprensible para usuarios con experiencia en MATLAB. Como consecuencia de esta búsqueda de equilibrio, el programa desarrollado sólo encuentra limitaciones con el aumento del tamaño de los conjuntos de datos que ha de procesar. Este problema se debe, principalmente, a la incapacidad de la memoria RAM de los ordenadores que comúnmente usan los investigadores, para gestionar sets de datos de varios gigabytes.

En la búsqueda de este equilibrio para la creación de un programa útil, contar con el *feedback* de los investigadores ha sido decisivo. Las capacidades básicas del programa, junto con los modos de visualización auxiliares y las opciones sugeridas por los investigadores, han propiciado que las visualizaciones obtenidas con esta herramienta sean usadas regularmente por el grupo de investigación CFU (Center for Fast Ultrasound Imaging). De esta forma, los investigadores usan el programa para facilitar su trabajo diario y compartir resultados con sus compañeros. Que se haya llegado a esta situación puede considerarse una indicación importante del éxito del proyecto acometido, que será alcanzado en su totalidad si el programa sigue siendo usado regularmente en por investigadores en este campo.

B.1 Trabajo futuro

Con el objetivo de continuar trabajando en el camino dedicado a ofrecer una visualización que cubra en todo momento las necesidades de los investigadores en el campo, se ha de considerar la posibilidad de que nuevas funcionalidades sean implementadas en el programa. A continuación se sugieren potenciales avances para mejora de los resultados obtenidos en el presente proyecto.

Como ya se ha indicado anteriormente, el programa fue diseñado con el objetivo de facilitar la implementación de nuevas funcionalidades, permitiendo a los usuarios añadir nuevas funciones al programa si aparecen nuevas necesidades de visualización.

Dejando a un lado el desarrollo de posibles nuevas funcionalidades para la visualización, especialmente para las visualizaciones 3D (un campo emergente que tiene un largo camino por delante lleno de desafíos), el programa encuentra un potencial margen de mejora cuando se enfrenta a sets de datos grandes. De esta forma, los sets de datos de varios gigabytes exceden la capacidad de la memoria RAM de un ordenador normal, lo que impide su visualización.

Una vez detectada esta situación, el autor de este proyecto desarrolló un código para solventar este problema. El código consiste en un bucle que carga, procesa y guarda por separado cada *frame* de datos (de un tamaño procesable) de un set de datos de varios gigabytes. Tras procesar todos los *frames* se puede crear un vídeo o gif a partir de los mismos. El tiempo total empleado en todo el proceso será similar al que habría tardado el programa desarrollado en este proyecto si los ordenadores usados fueran capaces de soportar sets de

datos tan grandes. Sin embargo, el tiempo requerido en posteriores visualizaciones con este código será siempre el mismo, ya que no se economizará tiempo de procesado construyendo a partir de visualizaciones previas. Como puede deducirse, este código falla en cuanto a la posibilidad de ofrecer una rápida sucesión de visualizaciones del mismo set de datos. Por ello, esta funcionalidad no ha sido implementada en el programa de este proyecto, ya que esto implicaría perder la capacidad de ofrecer visualizaciones rápidas y flexibles, lo que se encuentra entre los objetivos del proyecto. Sin embargo, la idea tras el código para visualizar sets de datos de gran tamaño ha de ser considerada en futuras mejoras del software que persigan superar este problema inherente a sets de datos de gran tamaño.

Por otra parte, se ha tener en cuenta que una de las principales decisiones de diseño tomadas en este proyecto fue la de no proporcionar una interfaz gráfica. De esta forma, el código del programa se expone al usuario durante la ejecución, permitiendo al usuario leerlo si es necesario. Sin embargo, en caso de que los investigadores en el campo lo consideraran necesario, el desarrollo de una interfaz gráfica para los *scripts* haría el proceso de visualización aún más simple.

Teniendo esto en cuenta como perspectiva sobre las que construir el futuro del presente proyecto, el trabajo realizado ha de ser visto no sólo como un programa que funciona y permite a los investigadores visualizar sus avances, actualmente, si no también como una sólida base que puede ser adaptada para la obtención nuevos tipos de visualizaciones en el futuro.

APPENDIX **C**

Presupuesto

1) Ejecución Material

- Compra de ordenador personal (Software incluido) — 2.000 €
- Material de oficina — 200 €

2) Honorarios Proyecto

- 1.800 horas a 18 €/ hora — 32.400 €

3) Material fungible

- Gastos de impresión — 200 €
- Encuadernación — 100 €

4) Subtotal del presupuesto

- Subtotal Presupuesto — 34.900 €

5) I.V.A. aplicable

- 21% Subtotal Presupuesto — 7.329 €

6) Total presupuesto

- Total Presupuesto — 42.229 €

Madrid, 5 de Mayo de 2015
El Ingeniero Jefe de Proyecto

Fdo.: Carlos Moreno López
Ingeniero de Telecomunicación

APPENDIX **D**

Pliego de condiciones

Pliego de condiciones

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un *Software for the visualization of blood flow velocity vectors from an ultrasound scanner*. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1) La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2) El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3) En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4) La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5) Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6) El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7) Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a

sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8) Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9) Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10) Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11) Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12) Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13) El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14) Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15) La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16) La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17) La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18) Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19) El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20) Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21) El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22) Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23) Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1) La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2) La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3) Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4) En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5) En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6) Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7) Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8) Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9) Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10) La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11) La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12) El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.